МОСКОВСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ ПРИБОРОСТРОЕНИЯ И ИНФОРМАТИКИ

Кафедра “Персональные ЭВМ”

Халабия Р.Ф.

Организация вычислительных систем и сетей. Учебное пособие.

Москва
2000

Халабия Р.Ф.

Организация вычислительных систем и сетей.
Учебное пособие.

Москва
2000

АННОТАЦИЯ

Настоящее учебное пособие предназнвчено для подготовки студентов различных форм обучения по специальностям: 22.01, 21.03.

В нем рассмотрены вопросы организации современных вычислительных систем, структуры микропроцессоров различных классов, памяти, локальных сетей.

Данное учебное поссобие модет использоваться в курсах "Структура и эксплуатация ЭВМ", "Организация ЭВМ, комплексов и систем", "Микропроцессорные системы управления", "Сети ЭВМ и средства телекоммуникаций".

Автор: Р.Ф. Халабия

Рецензенты: проф., д.т.н. Михайлов Б.М.
проф., к.т.н. Степанова Т.А.
доцент, к.т.н. Зеленко Г.В.
проф., к.т.н. Рощин А.В.

Работа рассмотрена и одобрена на заседании кафедры ИТ-4 "Персональные ЭВМ и сети".

Зав. каф., проф., д.т.н. Михайлов Б.М.

Содержание

Введение

Достижение микроэлектронной технологии позволили значительно расширить возможности всех классов электронных вычислительных машин. Так, разработаны новые микропроцессорные вычислительные средства, являющиеся основой микроЭВМ и персональных ЭВМ. В связи с этим особенно большое внимание следует уделить принципам организации ЭВМ и вычислительных систем различных классов.

Под организацией системы понимается принцип, положенный в основу функционирования системы и правила взаимодействия элементов, обеспечивающие реализацию функций системы.

Система – совокупность взаимосвязанных элементов, объединенных в единое целое для достижения результатов определяемых ее назначением.

Элемент – минимальный неделимый объект, характеризуемый собственными свойствами. Важнейшим из свойств является его целостность. Неделимость элемента обусловлена неизбежным изменением его свойств в случае разделения на отдельные части. При этом переход на другой уровень глубины изучения допускают разложение элемента на составные части, в свою очередь обладающие собственными свойствами отличные от свойств самого элемента.

Функция системы - как целевое назначение, так и поведение системы без относительных составляющих ее элементом и их связям, то есть без определения того как устроена система.

Структура системы – упорядоченная совокупность элементов и связей между ними, обеспечивающих реализацию функции системы. Инженерной формой изображению структуры системы является схема, в которой элементы и связи между ними обозначены фигурами удобными для практического применения.

Комплекс – система элементов в их взаимосвязи.

Вычислительная система – совокупность аппаратных и программных средств, объединенных для решения поставленных перед системой задач и обладающих способностью обработку информации и формировать управляющие воздействия под программным воздействием.

Вычислительная сеть (сеть ЭВМ) – сеть передачи данных, в одном или нескольких узлах которой располагаются вычислительные машины. Сеть передачи данных состоит из нескольких узлов (станций), соединенных различными каналами связи.

Архитектура ВС – совокупность характеристик и параметров, определяющих функциональную и структурную организацию системы, структуру обрабатываемых данных и т.д.

Понятие архитектуры ВС охватывает комплекс общих вопросов ее построения, возможностями и свойствами системы, а не деталями технической реализации.

Качество системы – мера одного из свойств системы всегда имеющая количественный смысл, поэтому использование некоторого показателя предполагает наличие способа измерения или оценки его значения, а также проведения измерения соответствующего свойства системы.

Оптимальная система – система, соответствующая максимальному значению прямого (минимальному для инверсного) критерия эффективности на множестве возможных вариантов построения системы.

1. Характеристики и режимы работы ЭВМ

1.1.Основные характеристики ЭВМ

Появление любого нового направления в вычислительной технике определяется требованиями компьютерного рынка. Поэтому у разработчиков компьютеров нет одной единственной цели, одна из которых повышение основных характеристик:

- отношение стоимость/производительность;
- надежность и отказоустойчивость;
- масштабируемость;
- совместимость и мобильность программного обеспечения.

Большая универсальная вычислительная машина (мейнфрейм) или суперкомпьютер стоят дорого. Для достижения поставленных целей при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики. Суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM относятся именно к этой категории компьютеров. Другим крайним примером может служить низкостоимостная конструкция, где производительность принесена в жертву для достижения низкой стоимости. К этому направлению относятся персональные компьютеры различных клонов IBM PC. Между этими двумя крайними направлениями находятся конструкции, основанные на отношении стоимость/ производительность, в которых разработчики находят баланс между стоимостными параметрами и производительностью. Типичными примерами такого рода компьютеров являются миникомпьютеры и рабочие станции.

Для сравнения различных компьютеров между собой обычно используются стандартные методики измерения производительности. Эти методики позволяют разработчикам и пользователям использовать полученные в результате испытаний количественные показатели для оценки тех или иных технических решений, и в конце концов именно производительность и стоимость дают пользователю рациональную основу для решения вопроса, какой компьютер выбрать.

Важнейшей характеристикой вычислительных систем является **надежность.** Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры.

Отказоустойчивость - это такое свойство вычислительной системы, которое обеспечивает ей, как логической машине, возможность продолжения действий, заданных программой, после возникновения неисправностей. Введение отказоустойчивости требует избыточного аппаратного и программного обеспечения. Направления, связанные с предотвращением неисправностей и с отказоустойчивостью, - основные в проблеме надежности. Концепции параллельности и отказоустойчивости вычислительных систем естественным образом связаны между собой, поскольку в обоих случаях требуются дополнительные функциональные компоненты. Поэтому, собственно, на параллельных вычислительных системах достигается как наиболее высокая производительность, так и, во многих случаях, очень высокая надежность. Имеющиеся ресурсы избыточности в параллельных системах могут гибко использоваться как для повышения производительности, так и для повышения надежности. Структура многопроцессорных и многомашинных систем приспособлена к автоматической реконфигурации и обеспечивает возможность продолжения работы системы после возникновения неисправностей.

Следует помнить, что понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности систем является целостность хранимых в них данных.

Масштабируемость представляет собой возможность наращивания числа и мощности процессоров, объемов оперативной и внешней памяти и других ресурсов вычислительной системы. Масштабируемость должна обеспечиваться архитектурой и конструкцией компьютера, а также соответствующими средствами программного обеспечения.

Добавление каждого нового процессора в действительно масштабируемой системе должно давать прогнозируемое увеличение производительности и пропускной способности при приемлемых затратах. Одной из основных задач при построении масштабируемых систем является минимизация стоимости расширения компьютера и упрощение планирования. В идеале добавление процессоров к системе должно приводить к линейному росту ее производительности. Однако это не всегда так. Потери производительности могут возникать, например, при недостаточной пропускной способности шин из-за возрастания трафика между процессорами и основной памятью, а также между памятью и устройствами ввода/вывода. В действительности реальное увеличение производительности трудно оценить

заранее, поскольку оно в значительной степени зависит от динамики поведения прикладных задач.

Возможность масштабирования системы определяется не только архитектурой аппаратных средств, но зависит от заложенных свойств программного обеспечения. Масштабируемость программного обеспечения затрагивает все его уровни от простых механизмов передачи сообщений до работы с такими сложными объектами как мониторы транзакций и вся среда прикладной системы. В частности, программное обеспечение должно минимизировать трафик межпроцессорного обмена, который может препятствовать линейному росту производительности системы. Аппаратные средства (процессоры, шины и устройства ввода/вывода) являются только частью масштабируемой архитектуры, на которой программное обеспечение может обеспечить предсказуемый рост производительности. Важно понимать, что простой переход, например, на более мощный процессор может привести к перегрузке других компонентов системы. Это означает, что действительно масштабируемая система должна быть сбалансирована по всем параметрам.

Концепция программной совместимости впервые в широких масштабах была применена разработчиками системы IBM/360. Основная задача при проектировании всего ряда моделей этой системы заключалась в создании такой архитектуры, которая была бы одинаковой с точки зрения пользователя для всех моделей системы независимо от цены и производительности каждой из них. Огромные преимущества такого подхода, позволяющего сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели были быстро оценены как производителями компьютеров, так и пользователями и начиная с этого времени практически все фирмы-поставщики компьютерного оборудования взяли на вооружение эти принципы, поставляя серии совместимых компьютеров. Следует заметить, что со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность внесения радикальных изменений архитектуру и способы организации вычислительных систем.

В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств. Это объясняется прежде всего тем, что для конечного пользователя в конце концов важно программное обеспечение, позволяющее решить его задачи, а не выбор той или иной аппаратной платформы. Переход от однородных сетей программно совместимых компьютеров к построению неоднородных сетей, включающих компьютеры разных фирм-производителей, в корне изменил и точку зрения на саму сеть: из сравнительно простого средства обмена информацией она превратилась в средство интеграции отдельных ресурсов - мощную распределенную вычислительную систему, каждый элемент которой (сервер или рабочая

станция) лучше всего соответствует требованиям конкретной прикладной задачи.

Этот переход выдвинул ряд новых требований. Прежде всего такая вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач. Во-вторых, она должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т.е. обеспечивать мобильность программного обеспечения. В третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть. В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения сформировалась концепция открытых систем, представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы.

Одним из вариантов моделей открытой среды является модель OSE (Open System Environment), предложенная комитетом IEEE POSIX. На основе этой модели национальный институт стандартов и технологии США выпустил документ "Application Portability Profile (APP). The U.S. Government's Open System Environment Profile OSE/1 Version 2.0", который определяет рекомендуемые для федеральных учреждений США спецификации в области информационных технологий, обеспечивающие мобильность системного и прикладного программного обеспечения. Все ведущие производители компьютеров и программного обеспечения в США в настоящее время придерживаются требований этого документа.

1.2. Режимы работы ЭВМ

В современных ЭВМ можно выделить следующие режимы работы:

- однопрограммный;
- мультипрограммный;
- пакетной обработки;
- разделения во времени;
- диалоговый;
- режим реального времени.

1.2.1. Однопрограммный режим работы

Режим, при котором выполняется не более одной независимой программы. При таком режиме работы ЭВМ решение задачи начинается с загрузки программы в ОП, после чего ЭВМ последовательно выполняет команды программы. При этом в каждый момент времени работает одно ее устройство, в то время как остальные простаивают в ожидании окончания ранее начатого действия. Значительные потери рабочего времени ЭВМ связаны с

медленной работой устройства ввода-вывода по сравнению с работой быстродействующих устройств (АЛУ, ЦУУ, ОЗУ и т.д.).

1.2.2. Мультипрограммный режим работы

Режим, при котором в памяти ЭВМ хранится несколько программ и выполнение одной программы может быть прервано для перехода к выполнению другой с последующим возвратом к прерванной программе. При совместном выполнении нескольких программ простои оборудования уменьшаются, поскольку увеличивается вероятность того, что среди находящихся в ЭВМ программ имеется одна, готовая к использованию освободившегося оборудования. Для уменьшения простоев оборудования ЭВМ широко применяют метод организации параллельной работы устройства ЭВМ за счет совмещения различных операций при работе ЭВМ. В целях более эффективного использования ЭВМ организуют мультипрограммную обработку информации на ЭВМ так, чтобы ею параллельно выполнялись команды, относящиеся к различным и независимым программам.

Мультипрограммный режим повышает производительность ЭВМ за счет увеличения числа задач, решаемых ЭВМ в течение некоторого промежутка времени. При этом время решения отдельной задачи увеличивается по сравнению с временем решения ее в однопрограммном режиме.

1.2.3. Режим пакетной обработки

Для обеспечения мультипрограммной обработки информации необходимо наличие нескольких задач, ожидающих обработки. Для эффективной загрузки ЭВМ используется режим пакетной обработки данных. В этом режиме задачи (программы и данные), подготовляемые многими пользователями ЭВМ, собираются в пачки-пакеты. Пакет состоит из заданий (не более 15), относящимся ко многим задачам, обработка которых занимает не менее часа машинного времени.

Различают два режима пакетной обработки. В первом число задач, выполняемых одновременно, фиксируется, а во втором не фиксируется, но в процессе обработки пакета ЭВМ оно может изменяться пакета ЭВМ оно может изменяться динамически. Пакет, предварительно записанный на том или ином носителе информации, вводится в ОЗУ ЭВМ. Когда пакет загружен, ЭВМ выбирает на обработку несколько задач и начинает выполнять их мультипрограммном режиме. Когда решение одной группы задач пакета закончено, из него выбирается для обработки следующая группа, это продолжается до тех пор, пока не будет обработана последняя группа задач пакета. После этого в ЭВМ вводится новый пакет задач.

Пакетная обработка данных позволяет увеличить производительность ЭВМ и уменьшить стоимость машинной обработки информации.

1.2.4. Режим разделения времени

Этот режим обеспечивает непосредственный и одновременный доступ к ЭВМ некоторому количеству пользователей чаще всего с дистанционно удаленных пунктов (терминалов). Терминал – периферийное устройство, предназначенное для обслуживания одного человека, решающего задачи на ЭВМ.

Пользователи с помощью терминалов вводят в ЭВМ исходные данные и программы и получают результаты вычислений. ЭВМ предоставляет каждому активному терминалу квант времени, равный секундам и долям секунды. По истечении этого времени ЭВМ переходит к обслуживанию следующего пользователя. За некоторый период времени ЭВМ обслуживает всех пользователей. При достаточно высоком быстродействии ЭВМ у отдельного пользователя создается иллюзия непрерывного контакта с ЭВМ.

Разделение времени позволяет устранить потери машинного времени, связанные с вмешательством оператора в работу ЭВМ из-за сравнительно низкой его скорости реакции, необходимости выполнения им определенных действий вне ЭВМ и медленного ввода информации с пульта оператора. При мультипрограммной работе ЭВМ в промежуточные паузы работы одного оператора к ЭВМ имеют доступ другие, что позволяет обеспечить полную загрузку внутренних устройств ЭВМ и тем самым поднять эффективность ее работы.

Режим разделения времени совместим с режимом пакетной обработки данных, которая предусматривается в ЭВМ для решения задач в отдельные периоды времени, когда пользователи не загружают ЭВМ полностью.

1.2.5. Диалоговый режим работы

Режим (режим “запрос-ответ”), при котором все программы пользователей постоянно хранятся в памяти ЭВМ и пользователи имеют непосредственный доступ к ЭВМ. От пользователей в ЭВМ поступают входные данные и запросы с пультовых пишущих машинок или дисплеев. Ответ формируется по программе, соответствующей определенному запросу. Выбор допустимых запросов ограничен емкостью памяти. Каждый запрос имеет соответствующий приоритет и временные ограничения на срок обслуживания.

1.2.6. Режим работы в реальном масштабе времени

Режим, при котором ЭВМ управляет работой какого-либо объекта или технологического процесса. Особенностью работы в реальном масштабе времени является то, что, помимо арифметической и логической обработки, выполняется слежение за работой объекта или прохождение некоторого процесса. Реализация этого режима привела к усложнению устройств и программного обеспечения ЭВМ.

2. Классификация компьютеров по областям применения

2.1. Персональные компьютеры и рабочие станции

Персональные компьютеры (ПК) появились в результате эволюции миникомпьютеров при переходе элементной базы машин с малой и средней степенью интеграции на большие и сверхбольшие интегральные схемы. ПК, благодаря своей низкой стоимости, очень быстро завоевали хорошие позиции на компьютерном рынке и создали предпосылки для разработки новых программных средств, ориентированных на конечного пользователя. Это прежде всего - "дружественные пользовательские интерфейсы", а также проблемно-ориентированные среды и инструментальные средства для автоматизации разработки прикладных программ.

Миникомпьютеры стали прародителями и другого направления развития современных систем - 32-разрядных машин. Создание RISC-процессоров и микросхем памяти емкостью более 1 Мбит, привело к окончательному оформлению настольных систем высокой производительности, которые сегодня известны как рабочие станции. Первоначальная ориентация рабочих станций на профессиональных пользователей (в отличие от ПК, которые в начале ориентировались на самого широкого потребителя непрофессионала) привела к тому, что рабочие станции - это хорошо сбалансированные системы, в которых высокое быстродействие сочетается с большим объемом оперативной и внешней памяти, высокопроизводительными внутренними магистралями, высококачественной и быстродействующей графической подсистемой и разнообразными устройствами ввода/вывода. Это свойство выгодно отличает рабочие станции среднего и высокого класса от ПК и сегодня. Даже наиболее мощные IBM PC совместимые ПК не в состоянии удовлетворить возрастающие потребности систем обработки из-за наличия в их архитектуре ряда "узких мест".

Тем не менее быстрый рост производительности ПК на базе новейших микропроцессоров Intel в сочетании с резким снижением цен на эти изделия и развитием технологии локальных шин (VESA и PCI), позволяющей устранить многие "узкие места" в архитектуре ПК, делают современные персональные компьютеры весьма привлекательной альтернативой рабочим станциям. В свою очередь производители рабочих станций создали изделия так называемого "начального уровня", которые по стоимостным характеристикам близки к высокопроизводительным ПК, но все еще сохраняют лидерство по производительности и возможностям наращивания.

Современный рынок "персональных рабочих станций" не просто определить. По сути он представляет собой совокупность архитектурных платформ персональных компьютеров и рабочих станций, которые появились в настоящее время, поскольку поставщики компьютерного оборудования

уделяют все большее внимание рынку продуктов для коммерции и бизнеса. Этот рынок традиционно считался вотчиной миникомпьютеров и мейнфреймов, которые поддерживали работу настольных терминалов с ограниченным интеллектом. В прошлом персональные компьютеры не были достаточно мощными и не располагали достаточными функциональными возможностями, чтобы служить адекватной заменой подключенных к главной машине терминалов. С другой стороны, рабочие станции на платформе UNIX были очень сильны в научном, техническом и инженерном секторах и были почти также неудобны, как и ПК для того чтобы выполнять серьезные офисные приложения. С тех пор ситуация изменилась коренным образом. Персональные компьютеры в настоящее время имеют достаточную производительность, а рабочие станции на базе UNIX имеют программное обеспечение, способное выполнять большинство функций, которые стали ассоциироваться с понятием "персональной рабочей станции". Вероятно оба этих направления могут серьезно рассматриваться в качестве сетевого ресурса для систем масштаба предприятия. В результате этих изменений практически ушли со сцены старомодные миникомпьютеры с их патентованной архитектурой и использованием присоединяемых к главной машине терминалов. По мере продолжения процесса разукрупнения (downsizing) и увеличения производительности платформы Intel наиболее мощные ПК (но все же чаще открытые системы на базе UNIX) стали использоваться в качестве серверов, постепенно заменяя миникомпьютеры.

Среди других факторов, способствующих этому процессу, следует выделить:

Применение ПК стало более разнообразным. Помимо обычных для этого класса систем текстовых процессоров, даже средний пользователь ПК может теперь работать сразу с несколькими прикладными пакетами, включая электронные таблицы, базы данных и высококачественную графику.

Адаптация графических пользовательских интерфейсов существенно увеличила требования пользователей ПК к соотношению производительность/стоимость. И хотя оболочка MS Windows может работать на моделях ПК 386SX с 2 Мбайтами оперативной памяти, реальные пользователи хотели бы использовать все преимущества подобных систем, включая возможность комбинирования и эффективного использования различных пакетов.

Широкое распространение систем мультимедиа прямо зависит от возможности использования высокопроизводительных ПК и рабочих станций с адеквантными аудио- и графическими средствами, и объемами оперативной и внешней памяти.

Слишком высокая стоимость мейнфреймов и даже систем среднего класса помогла сместить многие разработки в область распределенных систем и систем клиент-сервер, которые многим представляются вполне оправданной по

экономическим соображениям альтернативой. Эти системы прямо базируются на высоконадежных и мощных рабочих станциях и серверах.

В начале представлялось, что необходимость сосредоточения высокой мощности на каждом рабочем месте приведет к переходу многих потребителей ПК на UNIX-станции. Это определялось частично тем, что RISC-процессоры, использовавшиеся в рабочих станциях на базе UNIX, были намного более производительными по сравнению с CISC-процессорами, применявшимися в ПК, а частично мощностью системы UNIX по сравнению с MS-DOS и даже OS/2.

Производители рабочих станций быстро отреагировали на потребность в низкостоимостных моделях для рынка коммерческих приложений. Потребность в высокой мощности на рабочем столе, объединенная с желанием поставщиков UNIX-систем продавать как можно больше своих изделий, привела такие компании как Sun Microsystems и Hewlett Packard на рынок рабочих станций для коммерческих приложений. И хотя значительная часть систем этих фирм все еще ориентирована на технические приложения, наблюдается беспрецедентный рост продаж продукции этих компаний для работы с коммерческими приложениями, требующими все большей и большей мощности для реализации сложных, сетевых прикладных систем, включая системы мультимедиа.

2.2. X-терминалы

X-терминалы представляют собой комбинацию бездисковых рабочих станций и стандартных ASCII-терминалов. Бездисковые рабочие станции часто применялись в качестве дорогих дисплеев и в этом случае не полностью использовали локальную вычислительную мощь. Одновременно многие пользователи ASCII-терминалов хотели улучшить их характеристики, чтобы получить возможность работы в многооконной системе и графические возможности. Совсем недавно, как только стали доступными очень мощные графические рабочие станции, появилась тенденция применения "подчиненных" X-терминалов, которые используют рабочую станцию в качестве локального сервера.

На компьютерном рынке X-терминалы занимают промежуточное положение между персональными компьютерами и рабочими станциями. Поставщики X-терминалов заявляют, что их изделия более эффективны в стоимостном выражении, чем рабочие станции высокого ценового класса, и предлагают увеличенный уровень производительности по сравнению с персональными компьютерами. Быстрое снижение цен, прогнозируемое иногда в секторе X-терминалов, в настоящее время идет очевидно благодаря обострившейся конкуренции в этом секторе рынка. Многие компании начали активно конкурировать за распределение рынка, а быстрый рост объемных поставок создал предпосылки для создания такого рынка. В настоящее время

уже достигнута цена в $1000 для X-терминалов начального уровня, что делает эту технологию доступной для широкой пользовательской базы.

Как правило, стоимость X-терминалов составляет около половины стоимости сравнимой по конфигурации бездисковой машины и примерно четверть стоимости полностью оснащенной рабочей станции.

Типовой X-терминал включает следующие элементы:

- экран высокого разрешения - обычно размером от 14 до 21 дюйма по диагонали;
- микропроцессор на базе Motorola 68xxx или RISC-процессор типа Intel i960, MIPS R3000 или AMD29000;
- отдельный графический сопроцессор в дополнение к основному процессору, поддерживающий двухпроцессорную архитектуру, которая обеспечивает более быстрое рисование на экране и прокручивание экрана;
- базовые системные программы, на которых работает система X-Windows и выполняются сетевые протоколы;
- программное обеспечение сервера X11;
- переменный объем локальной памяти (от 2 до 8 Мбайт) для дисплея, сетевого интерфейса, поддерживающего TCP/IP и другие сетевые протоколы;
- порты для подключения клавиатуры и мыши.

X-терминалы отличаются от ПК и рабочих станций не только тем, что не выполняет функции обычной локальной обработки. Работа X-терминалов зависит от главной (хост) системы, к которой они подключены посредством сети. Для того, чтобы X-терминал мог работать, пользователи должны инсталлировать программное обеспечение многооконного сервера X11 на главном процессоре, выполняющим прикладную задачу (наиболее известная версия X11 Release 5). X-терминалы отличаются также от стандартных алфавитно-цифровых ASCII и традиционных графических дисплейных терминалов тем, что они могут быть подключены к любой главной системе, которая поддерживает стандарт X-Windows. Более того, локальная вычислительная мощь X-терминала обычно используется для обработки отображения, а не обработки приложений (называемых клиентами), которые выполняются удаленно на главном компьютере (сервере). Вывод такого удаленного приложения просто отображается на экране X-терминала.

Минимальный объем требуемой для работы памяти X-терминала составляет 1 Мбайт, но чаще 2 Мбайта. В зависимости от функциональных возможностей изделия оперативная память может расширяться до 32 Мбайт и более.

Оснащенный стандартной системой X-Windows, X-терминал может отображать на одном и том же экране множество приложений одновременно. Каждое приложение может выполняться в своем окне и пользователь может

изменять размеры окон, их месторасположение и манипулировать ими в любом месте экрана.

X-Windows - результат совместной работы Масачусетского технологического института (MIT) и корпорации DEC. Система X-Windows (известная также под именем X) в настоящее время является открытым де-факто стандартом для доступа к множеству одновременно выполняющихся приложений с возможностями многооконного режима и графикой высокого разрешения на интеллектуальных терминалах, персональных компьютерах, рабочих станциях и X-терминалах. Она стала стандартом для обеспечения интероперабельности (переносимости) продуктов многих поставщиков и для организации доступа к множеству приложений. В настоящее время X-Windows является стандартом для разработки пользовательского интерфейса. Более 90% поставщиков UNIX-рабочих станций и многие поставщики персональных компьютеров адаптировали систему X-Windows и применяют в качестве стандарта.

2.3. Серверы

Прикладные многопользовательские коммерческие бизнес-системы, включающие системы управления базами данных и обработки транзакций, крупные издательские системы, сетевые приложения и системы обслуживания коммуникаций, разработку программного обеспечения и обработку изображений все более настойчиво требуют перехода к модели вычислений "клиент-сервер" и распределенной обработке. В распределенной модели "клиент-сервер" часть работы выполняет сервер, а часть пользовательский компьютер (в общем случае клиентская и пользовательская части могут работать и на одном компьютере). Существует несколько типов серверов, ориентированных на разные применения: файл-сервер, сервер базы данных, принт-сервер, вычислительный сервер, сервер приложений. Таким образом, тип сервера определяется видом ресурса, которым он владеет (файловая система, база данных, принтеры, процессоры или прикладные пакеты программ).

С другой стороны существует классификация серверов, определяющаяся масштабом сети, в которой они используются: сервер рабочей группы, сервер отдела или сервер масштаба предприятия (корпоративный сервер). Эта классификация весьма условна. Например, размер группы может меняться в диапазоне от нескольких человек до нескольких сотен человек, а сервер отдела обслуживать от 20 до 150 пользователей. Очевидно в зависимости от числа пользователей и характера решаемых ими задач требования к составу оборудования и программного обеспечения сервера, к его надежности и производительности сильно варьируются.

Файловые серверы небольших рабочих групп (не более 20-30 человек) проще всего реализуются на платформе персональных компьютеров и программном обеспечении Novell NetWare. Файл-сервер, в данном случае, выполняет роль центрального хранилища данных. Серверы прикладных систем

и высокопроизводительные машины для среды "клиент-сервер" значительно отличаются требованиями к аппаратным и программным средствам.

Скорость процессора для серверов с интенсивным вводом/выводом некритична. Они должны быть оснащены достаточно мощными блоками питания для возможности установки дополнительных плат расширения и дисковых накопителей. Желательно применение устройства бесперебойного питания. Оперативная память обычно имеет объем не менее 32 Мбайт, что позволит операционной системе (например, NetWare) использовать большие дисковые кэши и увеличить производительность сервера. Как правило, для работы с многозадачными операционными системами такие серверы оснащаются интерфейсом SCSI (или Fast SCSI). Распределение данных по нескольким жестким дискам может значительно повысить производительность.

Наличии одного сегмента сети и 10-20 рабочих станций пиковая пропускная способность сервера ограничивается максимальной пропускной способностью сети. В этом случае замена процессоров или дисковых подсистем более мощными не увеличивают производительность, так как узким местом является сама сеть. Поэтому важно использовать хорошую плату сетевого интерфейса.

Однако для файл-серверов общего доступа, с которыми одновременно могут работать несколько десятков, а то и сотен человек, простой однопроцессорной платформы и программного обеспечения Novell может оказаться недостаточно. В этом случае используются мощные многопроцессорные серверы с возможностями наращивания оперативной памяти до нескольких гигабайт, дискового пространства до сотен гигабайт, быстрыми интерфейсами дискового обмена (типа Fast SCSI-2, Fast&Wide SCSI-2 и Fiber Channel) и несколькими сетевыми интерфейсами. Эти серверы используют операционную систему UNIX, сетевые протоколы TCP/IP и NFS. На базе многопроцессорных UNIX-серверов обычно строятся также серверы баз данных крупных информационных систем, так как на них ложится основная нагрузка по обработке информационных запросов. Подобного рода серверы получили название суперсерверов.

По уровню общесистемной производительности, функциональным возможностям отдельных компонентов, отказоустойчивости, а также в поддержке многопроцессорной обработки, системного администрирования и дисковых массивов большой емкости суперсерверы вышли в настоящее время на один уровень с мейнфреймами и мощными миникомпьютерами. Современные суперсерверы характеризуются:

- наличием двух или более центральных процессоров RISC, либо Pentium;
- многоуровневой шинной архитектурой, в которой запатентованная высокоскоростная системная шина связывает между собой несколько процессоров и оперативную память, а также множество стандартных шин ввода/вывода, размещенных в том же корпусе;

- поддержкой технологии дисковых массивов RAID;
- поддержкой режима симметричной многопроцессорной обработки, которая позволяет распределять задания по нескольким центральным процессорам или режима асимметричной многопроцессорной обработки, которая допускает выделение процессоров для выполнения конкретных задач.

Как правило, суперсерверы работают под управлением операционных систем UNIX, а в последнее время и Windows NT (на Digital 2100 Server Model A500MP), которые обеспечивают многопотоковую многопроцессорную и многозадачную обработку. Суперсерверы должны иметь достаточные возможности наращивания дискового пространства и вычислительной мощности, средства обеспечения надежности хранения данных и защиты от несанкционированного доступа. Кроме того, в условиях быстро растущей организации, важным условием является возможность наращивания и расширения уже существующей системы.

2.4. Мейнфреймы

Мейнфрейм - это синоним понятия "большая универсальная ЭВМ". Мейнфреймы и до сегодняшнего дня остаются наиболее мощными (не считая суперкомпьютеров) вычислительными системами общего назначения, обеспечивающими непрерывный круглосуточный режим эксплуатации. Они могут включать один или несколько процессоров, каждый из которых, в свою очередь, может оснащаться векторными сопроцессорами (ускорителями операций с суперкомпьютерной производительностью). В нашем сознании мейнфреймы все еще ассоциируются с большими по габаритам машинами, требующими специально оборудованных помещений с системами водяного охлаждения и кондиционирования. Однако это не совсем так. Прогресс в области элементно-конструкторской базы позволил существенно сократить габариты основных устройств. Наряду со сверхмощными мейнфреймами, требующими организации двухконтурной водяной системы охлаждения, имеются менее мощные модели, для охлаждения которых достаточно принудительной воздушной вентиляции, и модели, построенные по блочно-модульному принципу и не требующие специальных помещений и кондиционеров.

Основными поставщиками мейнфреймов являются известные компьютерные компании IBM, Amdahl, ICL, Siemens Nixdorf и некоторые другие, но ведущая роль принадлежит безусловно компании IBM. Именно архитектура системы IBM/360, выпущенной в 1964 году, и ее последующие поколения стали образцом для подражания. В нашей стране в течение многих лет выпускались машины ряда ЕС ЭВМ, являвшиеся отечественным аналогом этой системы.

В архитектурном плане мейнфреймы представляют собой многопроцессорные системы, содержащие один или несколько центральных и

периферийных процессоров с общей памятью, связанных между собой высокоскоростными магистралями передачи данных. При этом основная вычислительная нагрузка ложится на центральные процессоры, а периферийные процессоры (в терминологии IBM - селекторные, блок-мультиплексные, мультиплексные каналы и процессоры телеобработки) обеспечивают работу с широкой номенклатурой периферийных устройств.

Первоначально мейнфреймы ориентировались на централизованную модель вычислений, работали под управлением патентованных операционных систем и имели ограниченные возможности для объединения в единую систему оборудования различных фирм-поставщиков. Однако повышенный интерес потребителей к открытым системам, построенным на базе международных стандартов и позволяющим достаточно эффективно использовать все преимущества такого подхода, заставил поставщиков мейнфреймов существенно расширить возможности своих операционных систем в направлении совместимости. В настоящее время они демонстрирует свою "открытость", обеспечивая соответствие со спецификациями POSIX 1003.3, возможность использования протоколов межсоединений OSI и TCP/IP или предоставляя возможность работы на своих компьютерах под управлением операционной системы UNIX собственной разработки.

Стремительный рост производительности персональных компьютеров, рабочих станций и серверов создал тенденцию перехода с мейнфреймов на компьютеры менее дорогих классов: миникомпьютеры и многопроцессорные серверы. Эта тенденция получила название "разукрупнение" (downsizing). Однако этот процесс в самое последнее время несколько замедлился. Основной причиной возрождения интереса к мейнфреймам эксперты считают сложность перехода к распределенной архитектуре клиент-сервер, которая оказалась выше, чем предполагалось. Кроме того, многие пользователи считают, что распределенная среда не обладает достаточной надежностью для наиболее ответственных приложений, которой обладают мейнфреймы.

Очевидно выбор центральной машины (сервера) для построения информационной системы предприятия возможен только после глубокого анализа проблем, условий и требований конкретного заказчика и долгосрочного прогнозирования развития этой системы.

Главным недостатком мейнфреймов в настоящее время остается относительно низкое соотношение производительность/стоимость. Однако фирмами-поставщиками мейнфреймов предпринимаются значительные усилия по улучшению этого показателя.

Следует также помнить, что в мире существует огромная инсталлированная база мейнфреймов, на которой работают десятки тысяч прикладных программных систем. Отказаться от годами наработанного программного обеспечения просто не разумно. Поэтому в настоящее время ожидается рост продаж мейнфреймов по крайней мере до конца этого столетия. Эти системы, с одной стороны, позволят модернизировать существующие

системы, обеспечив сокращение эксплуатационных расходов, с другой стороны, создадут новую базу для наиболее ответственных приложений.

2.5. Кластерные архитектуры

Двумя основными проблемами построения вычислительных систем для критически важных приложений, связанных с обработкой транзакций, управлением базами данных и обслуживанием телекоммуникаций, являются обеспечение высокой производительности и продолжительного функционирования систем. Наиболее эффективный способ достижения заданного уровня производительности - применение параллельных масштабируемых архитектур. Задача обеспечения продолжительного функционирования системы имеет три составляющих: надежность, готовность и удобство обслуживания. Все эти три составляющих предполагают, в первую очередь, борьбу с неисправностями системы, порождаемыми отказами и сбоями в ее работе. Эта борьба ведется по всем трем направлениям, которые взаимосвязаны и применяются совместно.

Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры. Повышение уровня готовности предполагает подавление в определенных пределах влияния отказов и сбоев на работу системы с помощью средств контроля и коррекции ошибок, а также средств автоматического восстановления вычислительного процесса после проявления неисправности, включая аппаратурную и программную избыточность, на основе которой реализуются различные варианты отказоустойчивых архитектур. Повышение готовности есть способ борьбы за снижение времени простоя системы. Основные эксплуатационные характеристики системы существенно зависят от удобства ее обслуживания, в частности от ремонтопригодности, контролепригодности и т.д.

В последние годы в литературе по вычислительной технике все чаще употребляется термин "системы высокой готовности" (High Availability Systems). Все типы систем высокой готовности имеют общую цель - минимизацию времени простоя. Имеется два типа времени простоя компьютера: плановое и неплановое. Минимизация каждого из них требует различной стратегии и технологии. Плановое время простоя обычно включает время, принятое руководством, для проведения работ по модернизации системы и для ее обслуживания. Неплановое время простоя является результатом отказа системы или компонента. Хотя системы высокой готовности возможно больше ассоциируются с минимизацией неплановых простоев, они оказываются также полезными для уменьшения планового времени простоя.

Существует несколько типов систем высокой готовности, отличающиеся своими функциональными возможностями и стоимостью. Следует отметить, что высокая готовность не дается бесплатно. Стоимость систем высокой готовности на много превышает стоимость обычных систем. Вероятно, поэтому наибольшее распространение в мире получили кластерные системы, благодаря тому, что они обеспечивают достаточно высокий уровень готовности систем при относительно низких затратах. Термин "кластеризация" на сегодня в компьютерной промышленности имеет много различных значений. Строгое определение могло бы звучать так: "реализация объединения машин, представляющегося единым целым для операционной системы, системного программного обеспечения, прикладных программ и пользователей". Машины, кластеризованные вместе таким способом могут при отказе одного процессора очень быстро перераспределить работу на другие процессоры внутри кластера. Это, возможно, наиболее важная задача многих поставщиков систем высокой готовности.

Первой концепцию кластерной системы анонсировала компания DEC, определив ее как группу объединенных между собой вычислительных машин, представляющих собой единый узел обработки информации. По существу VAX-кластер представляет собой слабосвязанную многомашинную систему с общей внешней памятью, обеспечивающую единый механизм управления и администрирования. В настоящее время на смену VAX-кластерам приходят UNIX-кластеры. При этом VAX-кластеры предлагают проверенный набор решений, который устанавливает критерии для оценки подобных систем.

VAX-кластер обладает следующими свойствами:

- *Разделение ресурсов.* Компьютеры VAX в кластере могут разделять доступ к общим ленточным и дисковым накопителям. Все компьютеры VAX в кластере могут обращаться к отдельным файлам данных как к локальным.
- *Высокая готовность.* Если происходит отказ одного из VAX-компьютеров, задания его пользователей автоматически могут быть перенесены на другой компьютер кластера. Если в системе имеется несколько контроллеров внешних накопителей и один из них отказывает, другие контроллеры автоматически подхватывают его работу.
- *Высокая пропускная способность.* Ряд прикладных систем могут пользоваться возможностью параллельного выполнения заданий на нескольких компьютерах кластера.
- *Удобство обслуживания системы.* Общие базы данных могут обслуживаться с единственного места. Прикладные программы могут инсталлироваться только однажды на общих дисках кластера и разделяться между всеми компьютерами кластера.
- *Расширяемость.* Увеличение вычислительной мощности кластера достигается подключением к нему дополнительных VAX-

компьютеров. Дополнительные накопители на магнитных дисках и магнитных лентах становятся доступными для всех компьютеров, входящих в кластер.

Работа любой кластерной системы определяется двумя главными компонентами: высокоскоростным механизмом связи процессоров между собой и системным программным обеспечением, которое обеспечивает клиентам прозрачный доступ к системному сервису.

В настоящее время широкое распространение получила также технология параллельных баз данных. Эта технология позволяет множеству процессоров разделять доступ к единственной базе данных. Распределение заданий по множеству процессорных ресурсов и параллельное их выполнение позволяет достичь более высокого уровня пропускной способности транзакций, поддерживать большее число одновременно работающих пользователей и ускорить выполнение сложных запросов. Существуют три различных типа архитектуры, которые поддерживают параллельные базы данных:

- Симметричная многопроцессорная архитектура с общей памятью (Shared Memory SMP Architecture). Эта архитектура поддерживает единую базу данных, работающую на многопроцессорном сервере под управлением одной операционной системы. Увеличение производительности таких систем обеспечивается наращиванием числа процессоров, устройств оперативной и внешней памяти.
- Архитектура с общими (разделяемыми) дисками (Shared Disk Architecture). Это типичный случай построения кластерной системы. Эта архитектура поддерживает единую базу данных при работе с несколькими компьютерами, объединенными в кластер (обычно такие компьютеры называются узлами кластера), каждый из которых работает под управлением своей копии операционной системы. В таких системах все узлы разделяют доступ к общим дискам, на которых собственно и располагается единая база данных. Производительность таких систем может увеличиваться как путем наращивания числа процессоров и объемов оперативной памяти в каждом узле кластера, так и посредством увеличения количества самих узлов.
- Архитектура без разделения ресурсов (Shared Nothing Architecture). Как и в архитектуре с общими дисками, в этой архитектуре поддерживается единый образ базы данных при работе с несколькими компьютерами, работающими под управлением своих копий операционной системы. Однако в этой архитектуре каждый узел системы имеет собственную оперативную память и собственные диски, которые не разделяются между отдельными узлами системы. Практически в таких системах разделяется только общий коммуникационный канал между узлами системы. Производительность таких систем может увеличиваться путем

добавления процессоров, объемов оперативной и внешней (дисковой) памяти в каждом узле, а также путем наращивания количества таких узлов.

Таким образом, среда для работы параллельной базы данных обладает двумя важными свойствами: высокой готовностью и высокой производительностью. В случае кластерной организации несколько компьютеров или узлов кластера работают с единой базой данных. В случае отказа одного из таких узлов, оставшиеся узлы могут взять на себя задания, выполнявшиеся на отказавшем узле, не останавливая общий процесс работы с базой данных. Поскольку логически в каждом узле системы имеется образ базы данных, доступ к базе данных будет обеспечиваться до тех пор, пока в системе имеется по крайней мере один исправный узел. Производительность системы легко масштабируется, т.е. добавление дополнительных процессоров, объемов оперативной и дисковой памяти, и новых узлов в системе может выполняться в любое время, когда это действительно требуется.

Параллельные базы данных находят широкое применение в системах обработки транзакций в режиме on-line, системах поддержки принятия решений и часто используются при работе с критически важными для работы предприятий и организаций приложениями, которые эксплуатируются по 24 часа в сутки.

3. Организация современного ПК

На быстродействие ПК влияет не только тип процессора, но и набор микросхем, помогающий этому процессору реализовать свои возможности. Обычно системные платы для процессоров Pentium строились на базе микросхем 430VX, 430 HX, 430 TX, а для процессоров Pentium Pro – 440 FX.

До сих пор системы, разработанные на базе процессора Pentium II, Celeron строились в основном с использованием чипсетов 440 FX, 440 LX и 440 BX, также чипсетов других фирм производителей SIS и VIA Apollo. Под процессора Pentium III разработаны в настоящее время свои наборы микросхем, например, Intel810, Intel810. Данные наборы микросхем совместимы не только с процессорами компании INTEL, но и другими компаниями, например, AMD и CYRIX.

Совместное применение нового процессора и старого набора микросхем обычно не дает реализовать процессору все его возможности.

Архитектура ПК на базе набор микросхем 440 LX

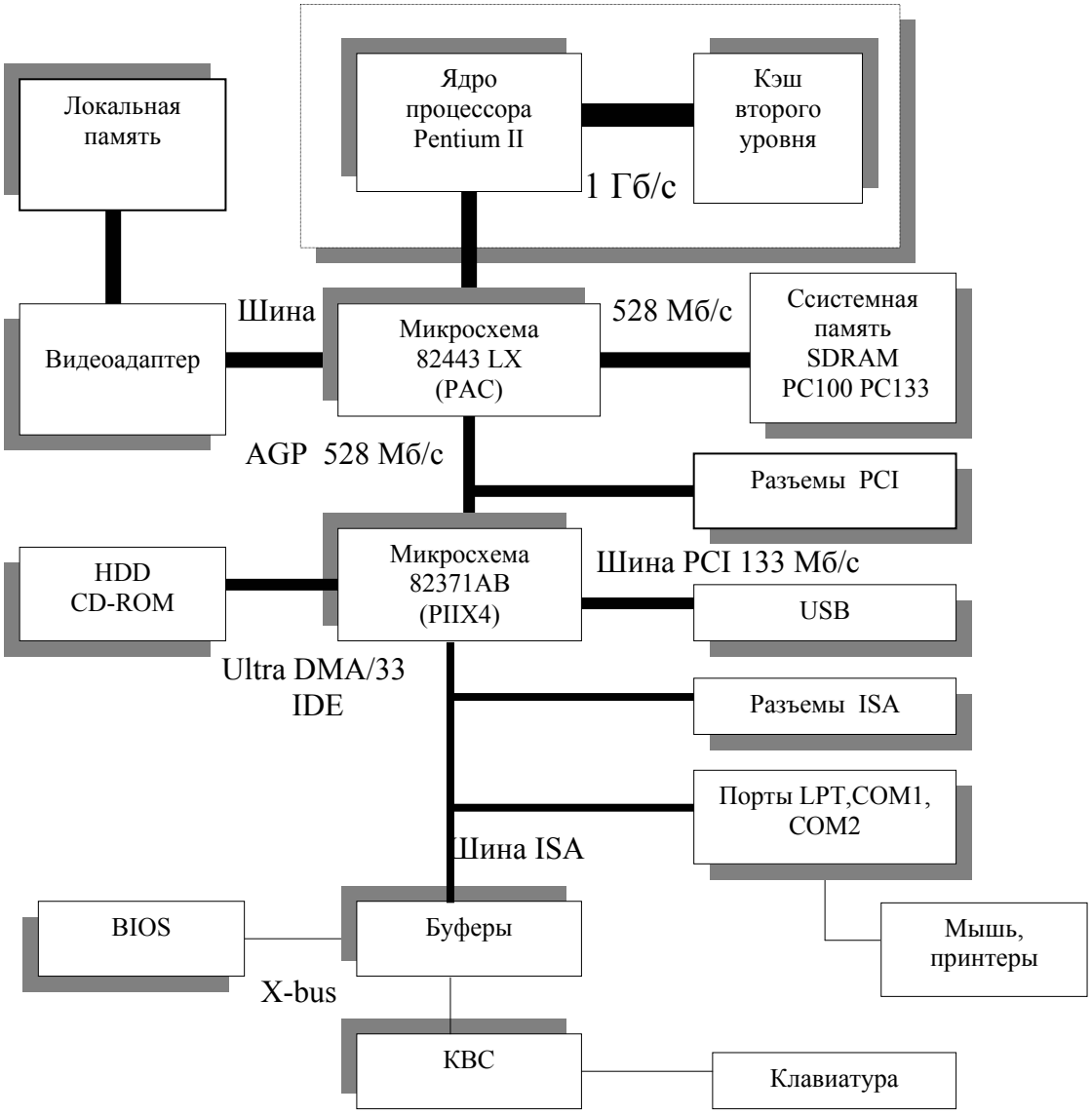


Рис. 3.1.

3.1. Структура системной платы на наборе микросхем 440 LX

Набор микросхем 440 LX позволяет разгрузить (132 Мб/с) шину PCI от жадного на ресурсы видеоадаптера и посадить последний на специально предназначенную для него (528 Мб/с) шину AGP.

Конструктивно набор состоит из двух устройств: выполненный в корпусе типа 492 BGA микросхемы 82443 LX (PAC или PCI AGP Controler) и заключенного в корпус типа 324 BGA многофункционального моста 82371 AB (PIIX4 или PCI, ISA, IDE Accelerator).

Чипсет 440 LX позволяет использовать синхронную динамическую память (SDRAM), которая подключается через шину с пропускной способностью 528 мб/с. На долю же PCI остаются более медленные устройства.

Микросхему 82443 LX проще всего представить в виде четырехпортового “черного ящика”, включающего в себя четыре буфера данных и четыре арбитражных устройства, регулирующих транспортные потоки блуждающих по системе байтов (см. рис. 3.1).

В целом же архитектуру ПК нового поколения можно сравнить с кровеносной системой, состоящей из сосудов трех диаметров: 1 Гб/с “аорты”, соединяющей ядро Pentium II с кеш-памятью второго уровня, трех более узких сосудов AGPset с ядром процессора, SDRAM и графическим акселератором, а также с системой PCI.

3.2. Типы системных плат на чипсете 440 LX

Применение набора увеличит быстродействие систем при выполнении целочисленных операций, действий с плавающей запятой и работе с мультимедия-приложеиями.

На данном чипсете комплектуются 3 типа системных плат:

AL 440LX – универсальной системной платы форм-фактора AX, предназначенной для домашних и корпоративных систем;

NX 440LX - высокоинтегрированной системной платы, созданной для снижения совокупной стоимости владения ПК в корпоративной среде;

DK 440LX – системные платы с двухпроцессорной конфигурацией на основе процессора Pentium II, разработанной для рабочих станций начального уровня и выполнения бизнес-приложений высокого уровня.

4. Функциональная и структурная организация процессоров

4.1. Классификация процессоров (CISC и RISC)

Двумя основными архитектурами набора команд, используемыми компьютерной промышленностью на современном этапе развития вычислительной техники являются архитектуры CISC и RISC. Основоположником CISC-архитектуры можно считать компанию IBM с ее базовой архитектурой /360, ядро которой используется с1964 года и дошло до наших дней, например, в таких современных мейнфреймах как IBM ES/9000.

Лидером в разработке микропроцессоров с полным набором команд (CISC - Complete Instruction Set Computer) считается компания Intel со своей серией x86 и Pentium. Эта архитектура является практическим стандартом для

рынка микрокомпьютеров. Для CISC-процессоров характерно: сравнительно небольшое число регистров общего назначения; большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов; большое количество методов адресации; большое количество форматов команд различной разрядности; преобладание двухадресного формата команд; наличие команд обработки типа регистр-память.

Основой архитектуры современных рабочих станций и серверов является архитектура компьютера с сокращенным набором команд (RISC - Reduced Instruction Set Computer). Зачатки этой архитектуры уходят своими корнями к компьютерам CDC6600, разработчики которых (Торнтон, Крэй и др.) осознали важность упрощения набора команд для построения быстрых вычислительных машин. Эту традицию упрощения архитектуры С. Крэй с успехом применил при создании широко известной серии суперкомпьютеров компании Cray Research. Однако окончательно понятие RISC в современном его понимании сформировалось на базе трех исследовательских проектов компьютеров: процессора 801 компании IBM, процессора RISC университета Беркли и процессора MIPS Стенфордского университета.

Разработка экспериментального проекта компании IBM началась еще в конце 70-х годов, но его результаты никогда не публиковались и компьютер на его основе в промышленных масштабах не изготавливался. В 1980 году Д.Паттерсон со своими коллегами из Беркли начали свой проект и изготовили две машины, которые получили названия RISC-I и RISC-II. Главными идеями этих машин было отделение медленной памяти от высокоскоростных регистров и использование регистровых окон. В 1981году Дж.Хеннесси со своими коллегами опубликовал описание стенфордской машины MIPS, основным аспектом разработки которой была эффективная реализация конвейерной обработки посредством тщательного планирования компилятором его загрузки.

Эти три машины имели много общего. Все они придерживались архитектуры, отделяющей команды обработки от команд работы с памятью, и делали упор на эффективную конвейерную обработку. Система команд разрабатывалась таким образом, чтобы выполнение любой команды занимало небольшое количество машинных тактов (предпочтительно один машинный такт). Сама логика выполнения команд с целью повышения производительности ориентировалась на аппаратную, а не на микропрограммную реализацию. Чтобы упростить логику декодирования команд использовались команды фиксированной длины и фиксированного формата.

Среди других особенностей RISC-архитектур следует отметить наличие достаточно большого регистрового файла (в типовых RISC-процессорах реализуются 32 или большее число регистров по сравнению с 8 - 16 регистрами в CISC-архитектурах), что позволяет большему объему данных храниться в регистрах на процессорном кристалле большее время и упрощает работу

компилятора по распределению регистров под переменные. Для обработки, как правило, используются трехадресные команды, что помимо упрощения дешифрации дает возможность сохранять большее число переменных в регистрах без их последующей перезагрузки.

Ко времени завершения университетских проектов (1983-1984 гг.) обозначился также прорыв в технологии изготовления сверхбольших интегральных схем. Простота архитектуры и ее эффективность, подтвержденная этими проектами, вызвали большой интерес в компьютерной индустрии и с 1986 года началась активная промышленная реализация архитектуры RISC. К настоящему времени эта архитектура прочно занимает лидирующие позиции на мировом компьютерном рынке рабочих станций и серверов.

Развитие архитектуры RISC в значительной степени определялось прогрессом в области создания оптимизирующих компиляторов. Именно современная техника компиляции позволяет эффективно использовать преимущества большего регистрового файла, конвейерной организации и большей скорости выполнения команд. Современные компиляторы используют также преимущества другой оптимизационной техники для повышения производительности, обычно применяемой в процессорах RISC: реализацию задержанных переходов и суперскалярной обработки, позволяющей в один и тот же момент времени выдавать на выполнение несколько команд.

Следует отметить, что в последних разработках компании Intel (имеется в виду Pentium P54C и процессор следующего поколения P6), а также ее последователей-конкурентов (AMD R5, Cyrix M1, NexGen Nx586 и др.) широко используются идеи, реализованные в RISC-микропроцессорах, так что многие различия между CISC и RISC стираются. Однако сложность архитектуры и системы команд x86 остается и является главным фактором, ограничивающим производительность процессоров на ее основе.

4.2.Принципы организации процессоров

4.2.1. Назначение и структура процессора

Процессор — центральная часть ЭВМ, организующая ее работу по заданной программе. Процессор объединяет в себе *АЛУ и ЦУУ,* с помощью которых рабочая программа интерпретируется в вычислительный процесс. Структура процессора зависит от принятой в ЭВМ системы счисления, формата данных и команд, системы команд, способов адресации и организации вычислительного процесса и принципа управления им, а также метода контроля и диагностики работы ЭВМ (см. рис. 4.1.).

Арифметическо-логическое устройство (АЛУ) — совокупность блоков и узлов процессора, обеспечивающая выполнение арифметических и логических операций над операндами. Характер операции задается командой программы.

Центральное устройство управления (ЦУУ) - совокупность блоков и узлов процессора, обеспечивающая координирование работы всех устройств

ЭВМ и управление ими для всех принятых в данной ЭВМ режимов работы. Процессор, АЛУ которого содержит один универсальный арифметическо-логический блок (АЛБ) для выполнения всех основных арифметических и логических операций, относят к процессорам универсального типа.

Процессор, АЛУ которого содержит несколько специализированных АЛБ, ориентированных на определенный тип выполняемых команд, относят к процессорам функционального типа.

По способу организации передачи и обработки информации различают процессоры последовательного, параллельного и параллельно-последовательного действия, по организации вычислительного процесса — на однопрограммные и мультипрограммные.

Процессоры последовательного и параллельно-последовательного действия применяют в тех случаях, когда к их быстродействию не предъявляют жестких требований. Модели процессоров имеют различные возможности для совмещения по времени работы отдельных функциональных блоков. Чем выше уровень совмещения, тем выше быстродействие процессора.

Структура микропроцессора

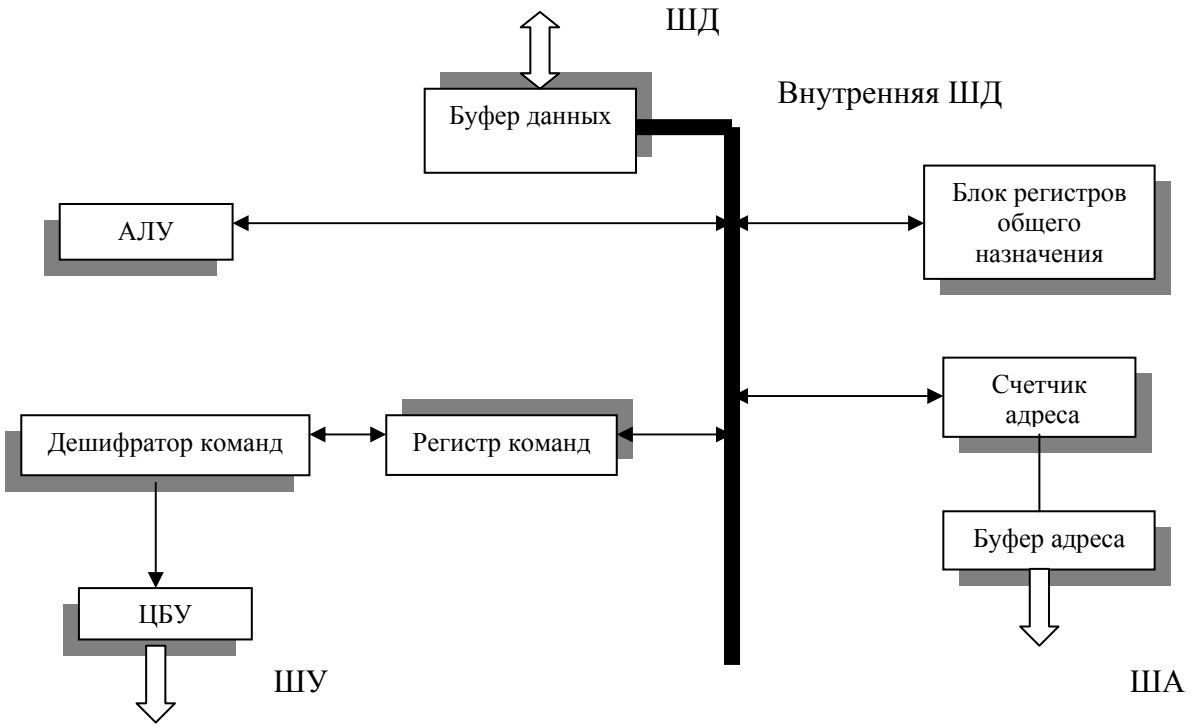


Рис. 4.1.

Cache (запас) - обозначает быстродействующую буферную память между процессором и основной памятью (буфер данных, буфер адреса). Кэш служит для частичной компенсации разницы в скорости процессора и основной памяти - туда попадают наиболее часто используемые данные. Когда процессор первый раз обращается к ячейки памяти, ее содержимое параллельно копируется в кэш,

и в случае повторного обращения в скором времени может быть с гораздо большей скоростью выбрано из кэша. При записи в память значение попадает в кэш, и либо одновременно копируется в память (схема Write Through - прямая или сквозная запись), либо копируется через некоторое время (схема Write Back - отложенная или обратная запись). При обратной записи, называемой также буферизованной сквозной записью, значение копируется в память в первом же свободном такте, а при отложенной (Delayed Write) - когда для помещения в кэш нового значения не оказывается свободной области; при этом в память вытесняются наименее используемая область кэша. Вторая схема более эффективна, но и более сложна за счет необходимости поддержания соответствия содержимого кэша и основной памяти.

4.2.2. Основные регистры процессоров

Обычно процессоры разделены на две части: операционное устройство (ОУ) и шинный интерфейс (ШИ). Роль ОУ заключается в выполнении команд, в то время как ШИ подготавливает команды и данные для выполнения. ОУ содержит АЛУ и устройство управления УУ и регистры общего назначения. Эти устройства обеспечивают выполнение команд, арифметические вычисления и логические операции (см. рис. 4.2).

Три элемента ШИ - блок управления шиной, очередь команд и сегментные регистры - осуществляют три важные функции:

1. ШИ управляет передачей данных на ОУ, в память и на внешние устройства ввода-вывода.
2. Сегментные регистры управляют адресацией памяти.
3. Выборка команд. Все программные команды находятся в памяти, и ШИ должен иметь доступ к ним для выборки их в очередь команд. ШИ должен "заглядывать вперед" и выбирать команды так, чтобы всегда существовала непустая очередь команд, готовых для выполнения.

ОУ и ШИ работают параллельно, причем ШИ опережает ОУ на один шаг. ОУ сообщает ШИ о необходимости доступа к данным в памяти или на устройство ввода-вывода. Кроме того, ОУ запрашивает машинные команды из очереди команд. Пока ОУ занято, ШИ выбирает следующую команду из памяти. Эта выборка происходит во время выполнения, что повышает скорость обработки.

Сегментные регистры

Сегментом называется область, которая начинается на границе параграфа, т.е. по любому адресу, кратному 16. Хотя сегмент может располагаться в любом месте памяти и иметь размер до 64 Кбайт, он требует столько памяти, сколько необходимо для выполнения программы.

Сегмент кодов (CS) содержит машинные команды, которые будут выполняться. Обычно первая выполняемая команда находится в начале этого

сегмента и операционная система передает управление по адресу данного сегмента для выполнения программы. Регистр сегмента кодов (CS) адресует данный сегмент.

Сегмент данных (DS) содержит определенные данные, константы и рабочие области, необходимые программе. Регистр сегмента данных (DS) адресует данный сегмент.

Сегмент стека (SS) содержит адреса возврата как для программы при возврате в операционную систему, так и для вызовов подпрограмм при возврате в главную программу. Регистр сегмента стека (SS) адресует данный сегмент.

Еще один сегментный регистр - дополнительный регистр сегмента (ES) - предназначен для специального использования.

Регистры общего назначения

Регистр - совокупность устройств, используемых для хранения информации, и обеспечения быстрого доступа к ней.

Регистр (AX) является основным сумматором и применяется для всех операций ввода-вывода, некоторых операций над строками и некоторых арифметических операций.

Регистр (BX) является базовым регистром. Это единственный регистр общего назначения, который может использоваться в качестве индекса для расширенной адресации.

Регистр (CX) является счетчиком. Он необходим для управления числом повторений циклов и для операций сдвига или вправо. Регистр (CX) используется также для вычислений.

Регистр (DX) является регистром данных. Он применяется для некоторых операций ввода-вывода и тех операций умножения и деления над большими числами.

Регистровые указатели (SP и BP) обеспечивают системе доступ к данным в сегменте стека. Регистр (SP) обеспечивает использование стека в памяти, позволяет временно хранить адреса и иногда данные. Этот регистр связан с регистром (SS) для адресации стека. Регистр (BP) облегчает доступ к параметрам (данным и адресам, переданным через стек).

Индексные регистры (SI и DI) применяются для расширенной адресации и для использования в операциях сложения и вычитания. Регистр (SI) является индексом источника и применяется для некоторых операций над строками. Регистр (DI) является индексом назначения и применяется также для некоторых операций над строками.

Регистр командного указателя (IP) содержит смещение на команду, которая должна быть выполнена.

Флаговый регистр определяет текущее состояние машины и результаты выполнения операций (проверка четности, переполнения, переносов, знака).

Операционное устройство и шинный интерфейс

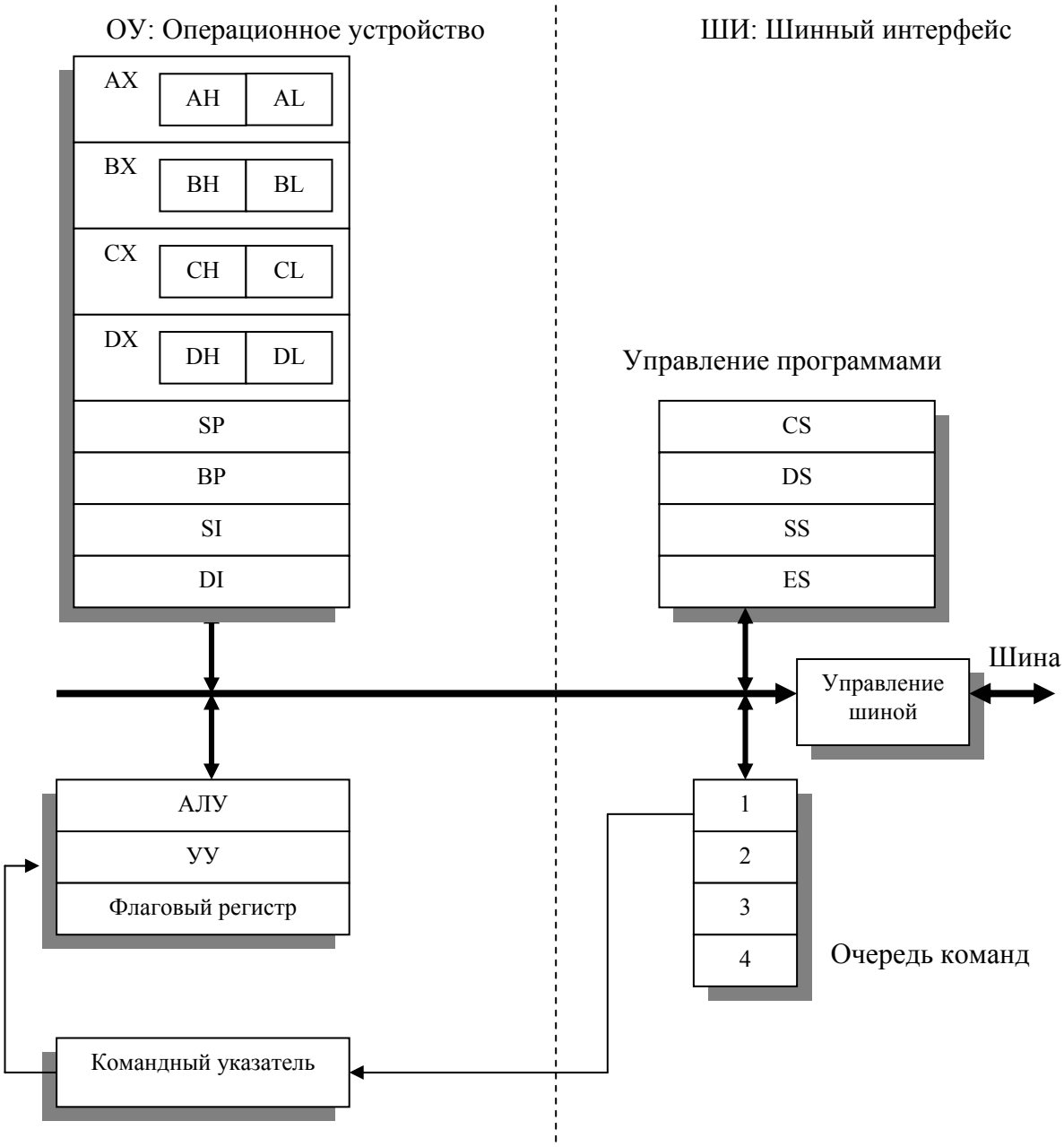


Рис. 4.2.

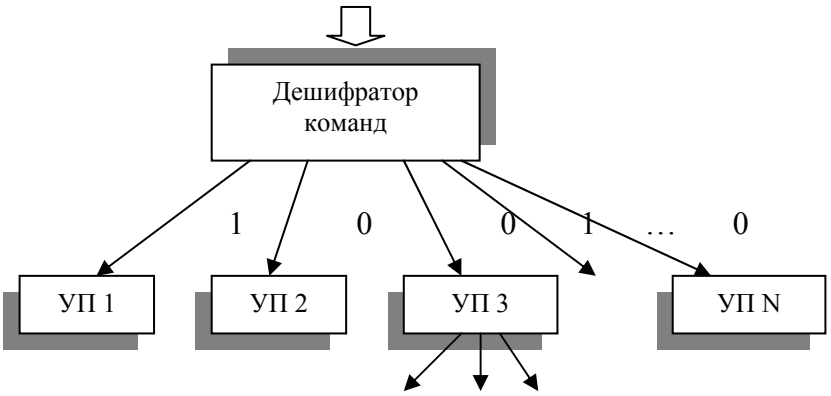
4.2.3. Способы организации управления вычислительным процессом

При автоматическом выполнении программы процессором команды последовательно поступают из оперативной памяти (ОП) в ЦУУ на время их выполнения АЛУ. Интервал времени, в течение которого процессор выполняет команду, называют рабочим циклом ЭВМ. Величина рабочего цикла зависит от структуры команды, типа операций, структуры операционных блоков АЛУ.

По принципу организации управления вычислительным процессом различают процессоры схемного типа или «жесткой» логикой, с микропрограммным и смешанным (микропрограммно-схемным) управлением.

Схемное управление - управление, при котором для выполнения любой операции последовательность управляющих сигналов задается логическими схемами (см. рис. 4.3). Различают центральное, местное и смешанное схемное управление.

Схемный принцип управления



УПn – управляющие части

Рис. 4.3.

В процессорах с центральным управлением длительность рабочего цикла выбирается такой, чтобы за время между двумя управляющими сигналами выполнялась самая длинная операция в процессоре. Такие процессоры получили название синхронных, а блок, в котором формируются управляющие сигналы для всех исполнительных устройств ЭВМ, называют центральным блоком управления (ЦБУ).

В синхронных процессорах при выполнении большинства операций, особенно коротких (например, операция сложения), происходит потеря машинного времени, связанная с непроизводительными простоями процессора. Однако структура процессора отличается простотой, экономичностью и удобна в эксплуатации.

В процессорах с местным управлением вычислительным процессом управление производится так, что каждая операция выполняется после выполнения предыдущей операции. При этом каждое исполнительное устройство после окончания работы формирует сигнал «Конец работы», который одновременно является сигналом «Начало работы» другого исполнительного устройства. Процессоры с переменной длительностью рабочего цикла, величина которого зависит от вида выполняемой операции и кодов операндов, называют асинхронными. В асинхронных процессорах основные исполнительные устройства имеют местные (автономные) блоки

управления, что резко повышает быстродействие таких процессоров, так как отсутствуют простои между реальными циклами выполнения команд. Основной недостаток асинхронных процессоров — их сложность.

В процессорах со смешанным управлением исполнение простейших операций осуществляется в синхронном режиме, а наиболее сложные операции (например, деление, умножение и др.) - в асинхронном. При смешанном управлении процессор содержит как центральный блок, так и местные блоки управления операциями. Смешанный способ управления вычислительным процессом позволяет получить высокое быстродействие процессора при умеренных затратах оборудования, а поэтому наиболее распространен в современных ЭВМ.

Микропрограммное управление (см. рис. 4.4) основано на замене управляющих логических схем (см. рис. 4.5) специальной программой, хранящейся в ПЗУ. При таком управлении каждая команда разделяется на ряд элементарных этапов, получивших название микроопераций. Последовательность микрокоманд, выполняющих одну команду (операцию), представляет собой микропрограмму. Для характеристики временных соотношений между различными этапами операции используется понятие машинный такт, определяющий интервал времени, в течение которого выполняется одна или одновременно несколько микроопераций.

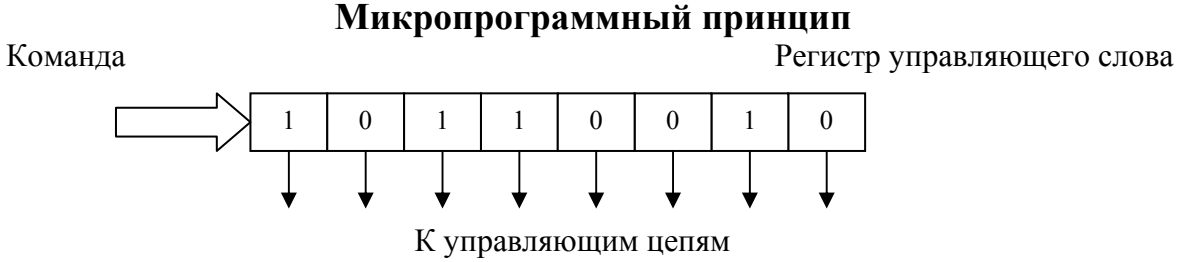
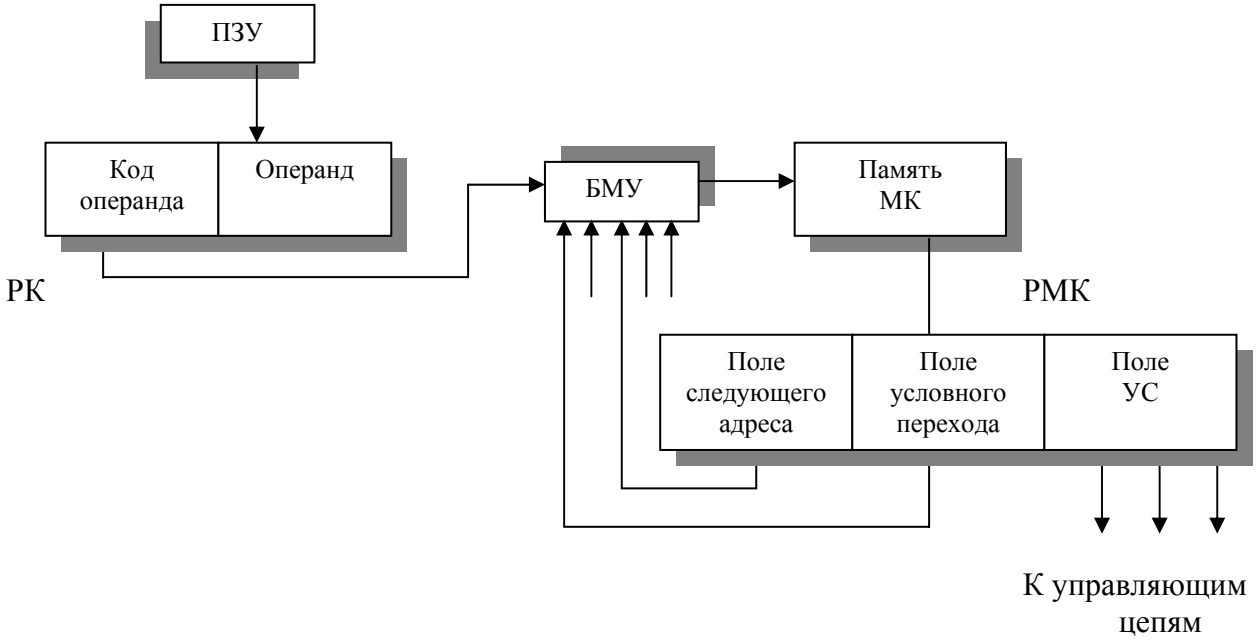


Рис. 4.4.

Достоинство микропрограммного управления заключается в том, что для изменения вида операций нет необходимости в переделке сложных электронных схем, неизбежной в ЭВМ со схемным управлением, а следует только изменить микропрограмму. Это обстоятельство дает возможность в данной ЭВМ использовать программы, составленные для другой ЭВМ. Благодаря этому микропрограммное управление получило широкое распространение в современных ЭВМ.

Схема управляющего устройства при микропрограммном принципе управления



БМУ – блок микропрограммного управления
МК - микрокоманда
РК - регистр команд
УС - управляющее слово
РМК – регистр микрокоманд

Рис. 4.5.

4.2.4. Технология MMX

Технология MMX - разработана для ускорения мультимедия и коммуникационных программ. Она включает в себя новые команды и типы данных, что позволяет создавать приложения нового уровня. Технология основана на параллельной обработке данных. При этом сохраняется полная совместимость с существующими операционными системами и программным обеспечением. MMX-технологии поддерживает новую арифметику, называемую арифметикой с насыщением (Saturation arithmetic).

Наибольший эффект от использования MMX-технологии может быть достигнут в алгоритмах со следующими характеристиками:

- малый размер данных (8-битные графические пикселы, 16-битные звуковые данные);
- короткие, часто повторяющиеся циклы;
- частые умножения и накопления.

В основе MMX лежит принцип SIMD (Single Instruction Multiple Data), т.е. одной командой можно обработать сразу несколько единиц информации.

Отображение MMX-регистров на FPU-регистры

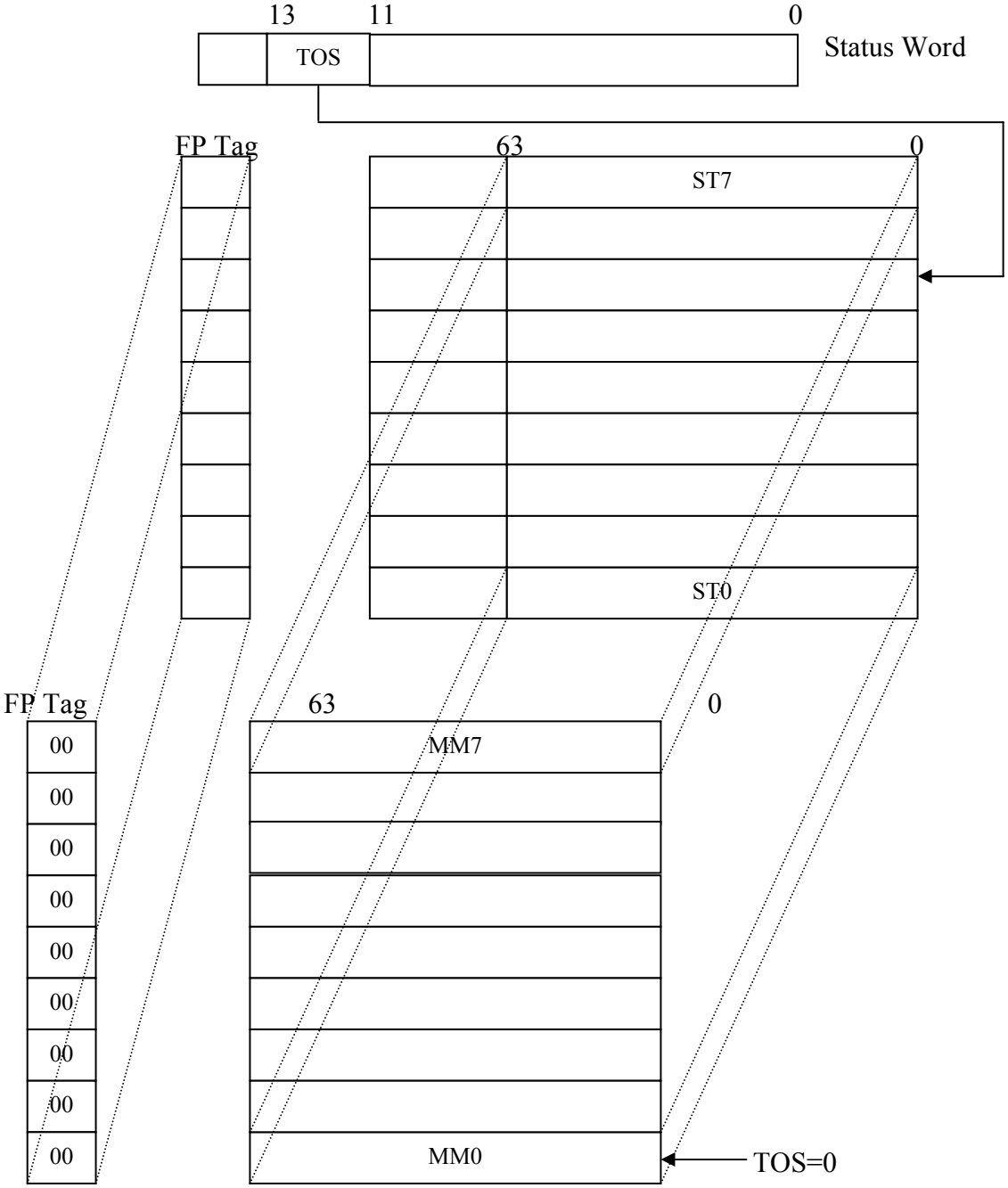


Рис. 4.6.

Технология MMX основана на отображении регистров MMX на регистры FPU (см. рис. 4.6). Главным образом это сделано для сохранения с существующим программным обеспечением.

Из рис. 4.6. видно, что MMX-регистры отображены на поля мантиссы в FPU-регистрах. Значение, записываемое в MMX-регистр, автоматически

появляется в младших битах (биты 63-0) соответствующих FPU-регистров. При этом в поле порядка (биты 78-64) и знаковый бит (бит 79) заносятся единицы. Значение поля TOS (Top Of Stack) устанавливается в нуль после выполнения каждой MMX-команды. Значение мантиссы, записываемое в FPU-регистр с помощью FPU-команды, автоматически появляется в соответствующем MMX-регистре.

Отображение MMX-регистров фиксировано и не зависит от значения поля TOS (биты 11-13 в регистре состояния FPU). В обозначении MMn, n - указывает на физический номер регистра, а в STn - n указывает на относительный номер регистра (относительно поля TOS).

При TOS=0: MM0 отображается на ST0, MM1 - ST1 и т.д.

При TOS=2: MM0 отображается на ST6, MM1 - ST6, MM2 - ST0 и т.д.

После выполнения любой MMX-команды (кроме EMMS) значения всех полей регистра тегов устанавливается в 00. Команда EMMS устанавливает значения всех полей регистра тегов 11 (см. табл.4.1.). Значения регистра тегов не оказывает никакого влияния на MMX-регистры или выполнения MMX-команд.

Так как MMX и FPU используют фактически и те же регистры, для сохранения и восстановления контекста MMX используются команды FSAVE (Store FP state) и FRSTOR (Restore FP state). Если при попытке выполнить MMX-команду бит TS в регистре CR0 установлен в единицу, то генерируется исключение Int7. Благодаря этому факту обеспечивается прозрачность управления контекстом MMX для операционной системы.

Таблица 4.1

Влияние MMX-команд на контекст FPU

Тип команды	**Регистр тегов**	**оле TOS**	**Другие регистры**	**Поле порядка и знаковый бит MMn (79…64)**	**Поле мантисы MMn (63…00)**
Чтение из MMX- регистра	Все поля 00	00	Не изменяется	Не изменяется	Не изменяется
Запись из MMX- регистра	Все поля 00	00	Не изменяется	Заполняется единицами	Переписывается
EMMS	Все поля 11	00	Не изменяются	Не изменяется	Не изменяется

4.2.5. Принципы конвейерной технологии

Разработчики архитектуры компьютеров издавна прибегали к методам проектирования, известным под общим названием "совмещение операций", при

котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции. Этот общий метод включает два понятия: параллелизм и конвейеризацию. Хотя у них много общего и их зачастую трудно различать на практике, эти термины отражают два совершенно различных подхода. При параллелизме совмещение операций достигается путем воспроизведения в нескольких копиях аппаратной структуры. Высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация (или *конвейерная обработка*) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Так обработку любой машинной команды можно разделить на несколько этапов (несколько ступеней), организовав передачу данных от одного этапа к следующему. При этом конвейерную обработку можно использовать для совмещения этапов выполнения разных команд. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

Для иллюстрации основных принципов построения процессоров мы будем использовать простейшую архитектуру, содержащую 32 целочисленных регистра общего назначения (R0,...,R31), 32 регистра плавающей точки (F0,...,F31) и счетчик команд PC. Будем считать, что набор команд нашего процессора включает типичные арифметические и логические операции, операции с плавающей точкой, операции пересылки данных, операции управления потоком команд и системные операции. В арифметических командах используется трехадресный формат, типичный для RISC-процессоров, а для обращения к памяти используются операции загрузки и записи содержимого регистров в память.

Выполнение типичной команды можно разделить на следующие этапы:

1. Выборка команды - IF (по адресу, заданному счетчиком команд, из памяти извлекается команда);
2. Декодирование команды / выборка операндов из регистров - ID;
3. Выполнение операции / вычисление эффективного адреса памяти - EX;
4. Обращение к памяти - MEM;
5. Запоминание результата - WB.

Работу конвейера можно условно представить в виде сдвинутых во времени схем процессора (рис. 4.7). Этот рисунок хорошо отражает совмещение во времени выполнения различных этапов команд.

Однако чаще для представления работы конвейера используются временные диаграммы (табл 4.2), на которых обычно изображаются выполняемые команды, номера тактов и этапы выполнения команд.

Конвейеризация увеличивает пропускную способность процессора (количество команд, завершающихся в единицу времени), но она не сокращает время выполнения отдельной команды. В действительности, она даже несколько увеличивает время выполнения каждой команды из-за накладных расходов, связанных с управлением регистровыми станциями. Однако увеличение пропускной способности означает, что программа будет выполняться быстрее по сравнению с простой неконвейерной схемой.

Тот факт, что время выполнения каждой команды в конвейере не уменьшается, накладывает некоторые ограничения на практическую длину конвейера. Кроме ограничений, связанных с задержкой конвейера, имеются также ограничения, возникающие в результате несбалансированности задержки на каждой его ступени и из-за накладных расходов на конвейеризацию. Частота синхронизации не может быть выше, а, следовательно, такт синхронизации не может быть меньше, чем время, необходимое для работы наиболее медленной ступени конвейера. Накладные расходы на организацию конвейера возникают из-за задержки сигналов в конвейерных регистрах (защелках) и из-за перекосов сигналов синхронизации. Конвейерные регистры к длительности такта добавляют время установки и задержку распространения сигналов. В предельном случае длительность такта можно уменьшить до суммы накладных расходов и перекоса сигналов синхронизации, однако при этом в такте не останется времени для выполнения полезной работы по преобразованию информации.

Конвейеризация эффективна только тогда, когда загрузка конвейера близка к полной, а скорость подачи новых команд и операндов соответствует максимальной производительности конвейера. Если произойдет задержка, то параллельно будет выполняться меньше операций и суммарная производительность снизится. Такие задержки могут возникать в результате возникновения конфликтных ситуаций.

Представление о работе конвейера

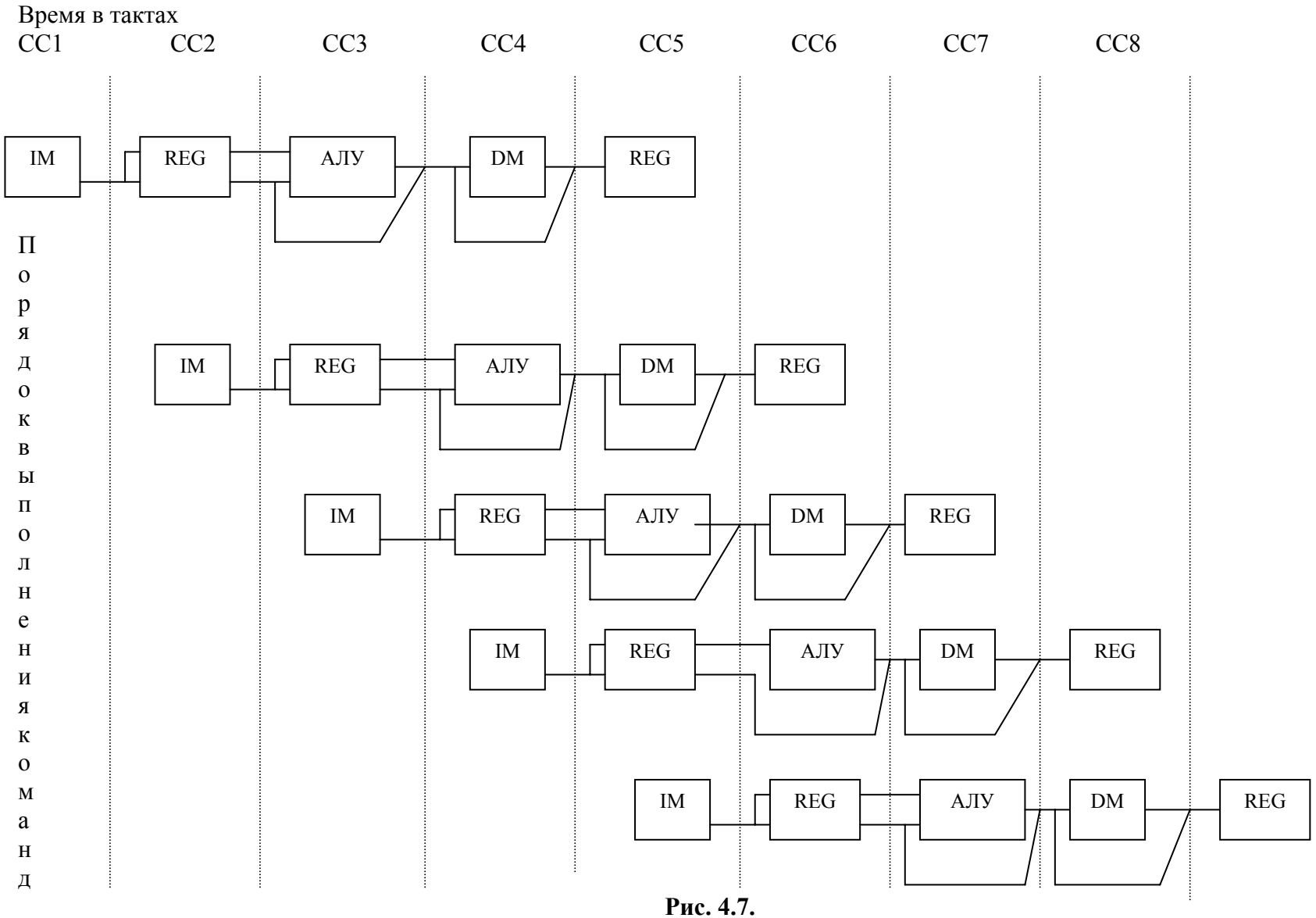


Рис. 4.7.

Таблица 4.2.

Диаграмма работы простейшего конвейера

Номер команды	**Номер такта**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Команда i	IF	ID	EX	MEM	WB				
Команда i+1		IF	ID	EX	MEM	WB			
Команда i+2			IF	ID	EX	MEM	WB		
Команда i+3				IF	ID	EX	MEM	WB	
Команда i+4					IF	ID	EX	MEM	WB

В качестве примера рассмотрим неконвейерную машину с пятью этапами выполнения операций, которые имеют длительность 50, 50, 60, 50 и 50 нс соответственно. Пусть накладные расходы на организацию конвейерной обработки составляют 5 нс. Тогда среднее время выполнения команды в неконвейерной машине будет равно 260 нс. Если же используется конвейерная организация, длительность такта будет равна длительности самого медленного этапа обработки плюс накладные расходы, т.е. 65 нс. Это время соответствует среднему времени выполнения команды в конвейере. Таким образом, ускорение, полученное в результате конвейеризации, будет равно:

$$\frac{\text{Среднее время выполнения команды в неконвейерном режиме}}{\text{Среднее время выполнения команды в конвейерном режиме}} = \frac{260}{65} = 4$$

Конвейеризация эффективна только тогда, когда загрузка конвейера близка к полной, а скорость подачи новых команд и операндов соответствует максимальной производительности конвейера. Если произойдет задержка, то параллельно будет выполняться меньше операций и суммарная производительность снизится. Такие задержки могут возникать в результате возникновения конфликтных ситуаций.

При реализации конвейерной обработки возникают ситуации, которые препятствуют выполнению очередной команды из потока команд в предназначенном для нее такте. Такие ситуации называются конфликтами. Конфликты снижают реальную производительность конвейера, которая могла бы быть достигнута в идеальном случае. Существуют три класса конфликтов:

1. Структурные конфликты, которые возникают из-за конфликтов по ресурсам, когда аппаратные средства не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением.
2. Конфликты по данным, возникающие в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды.

3. Конфликты по управлению, которые возникают при конвейеризации команд переходов и других команд, которые изменяют значение счетчика команд.

Конфликты в конвейере приводят к необходимости приостановки выполнения команд (pipeline stall). Обычно в простейших конвейерах, если приостанавливается какая-либо команда, то все следующие за ней команды также приостанавливаются. Команды, предшествующие приостановленной, могут продолжать выполняться, но во время приостановки не выбирается ни одна новая команда.

4.3. Микроархитектура процессоров P5

Процессор Pentium своей конвейерной и суперскалярной архитектурой достиг впечатляющего уровня производительности.

Pentium содержит два 5-стадийных конвейера, которые могут работать параллельно и выполнять две целочисленные команды за машинный такт. При этом параллельно может выполняться только пара команд, следующих в программе друг за другом и удовлетворяющих определенным правилам, например, отсутствие регистровых зависимостей типа "запись после чтения".

В Pentium для увеличения пропускной способности осуществлен переход к одному 12-стадийному конвейеру. Увеличение числа стадий приводит к уменьшению выполняемой на каждой стадии работы и, как следствие, к уменьшению времени нахождения команды на каждой стадии на 33 процента по сравнению с другими процессорами. Это означает, что использование при производстве Pentium той же технологии, что и при производстве 100 МГц Pentium, приведет к получению Pentium с тактовой частотой 133 МГц.

Возможности суперскалярной архитектуры Pentium, с ее способностью к выполнению двух команд за такт, было бы трудно превзойти без совершенно нового подхода. Примененный в Pentium новый подход устраняет жесткую зависимость между традиционными фазами "выборки" и "выполнения", когда последовательность прохождения команд через эти две фазы соответствует последовательности команд в программе.

На рис. 4.8 изображена схема процессора процессора P5, состоящего из 2 конвейеров, U-конвейера и V-конвейера. U-конвейер может выполнять все команды целого типа и команды с плавающей точкой. V-конвейер может выполнять простые целые команды и команды FXCH с плавающей точкой.

Кэш разделен на Кэш-данных и Кэш-команд. Кэш-данных имеет два порта, которые состоят из двух конвейеров. Кеш-данных имеет буфер преобразования таблицы страниц (TLB) который переводит линейный адрес в физический адрес используемый кешом-данных.

Кеш-команд, буфер переходов и буфер выборки отвечают за получение команд в исполняющем модуле процессора Команды выбираются из кеша-команд или с внешней шины. Адрес перехода запоминается в буфере переходов. TLB кеша команд транслирует линейный адрес в физический

Структурная схема микропроцессора Pentium

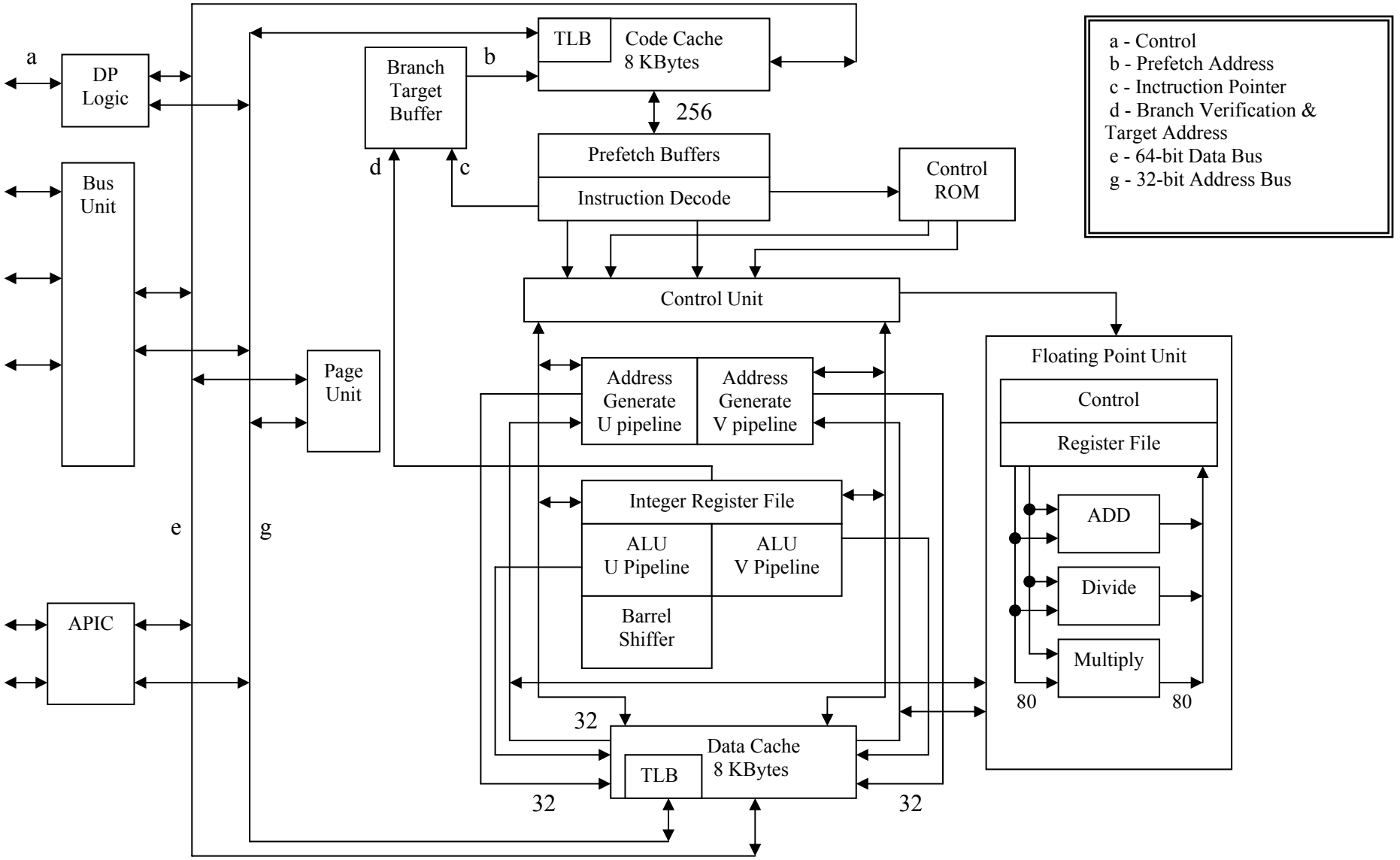


Рис. 4.8.

используемый кешом команд. Декодер перекодирует выбранные команды так , что процессор мог их выполнять. ROM контроль содержит микрокод с контролем последовательности операций, который может быть эффективен для реализации в архитектурой процессора. Контроль ROM имеет директиву контроля над обоими конвейерами.

Процессор Pentium содержит конвейер плавающей точки, который обеспечивает выполнение существенных эффективных предыдущих вычислений процессора.

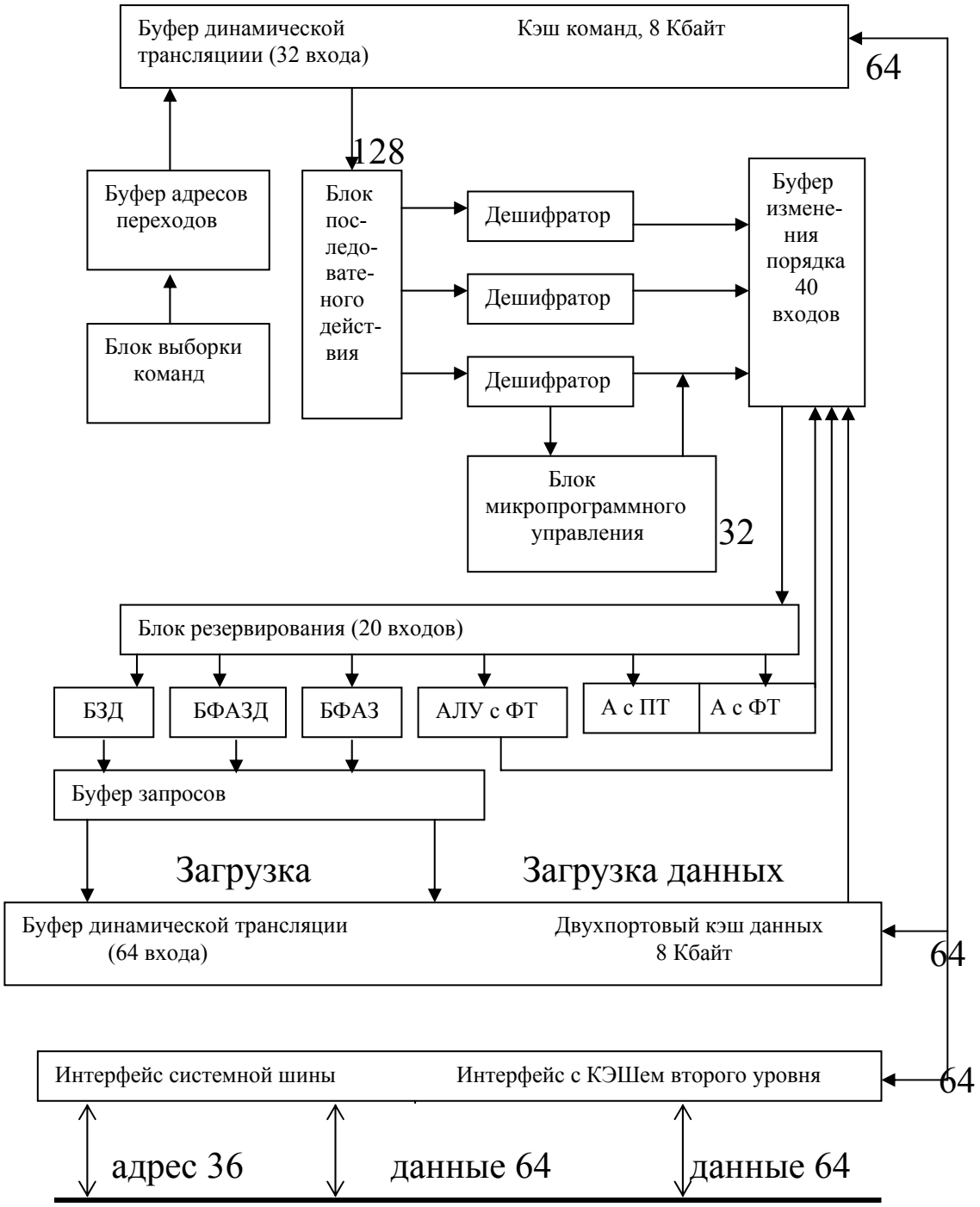
4.4. Микроархитектура процессоров семейства P6

Процессор Pentium представляет собой довольно простое и весьма ограниченное приближение к суперскалярной архитектуре. Два его конвейера нельзя считать полностью независимыми, поскольку в случае остановки одного из них другой также должен быть остановлен и, таким образом, динамический запуск команд на исполнение (т. е. запуск команд с нарушением последовательности их расположения в программе) не представляется возможным. Более того, блок арифметики с плавающей точкой не является полностью автономным, а связан с конвейерами целочисленных операций. Следовательно, инструкции целочисленной арифметики и арифметики с плавающей точкой не могут исполняться параллельно. Процессоры P6, напротив, можно смело отнести к ЦП с суперскалярной архитектурой, позволяющей формировать за один машинный такт сразу несколько результатов (естественно, спустя некоторое время, называемое временем старта. Кроме того, эти процессоры отличается повышенной тактовой частотой. На рис. 4.9 приведена блок-схема процессора Pentium Pro.

Разработчики Pentium Pro полностью развязали схемы диспетчеризации потока команд и их исполнения. Инструкции x86 преобразуются в последовательность внутренних микроопераций, по сути не отличающихся от традиционных микрокоманд. Затем эти микрокоманды поступают в буфер изменения порядка с 40 входами, где они хранятся до тех пор, пока не будут «готовы» к обработке необходимые операнды. Из этого буфера инструкции передаются в блок резервирования с 20 входами, где и ожидают, когда освободится необходимое операционное устройство. Подобная структура позволяет исполнять микрооперации с нарушением порядка и загружать одновременно несколько операционных устройств. Ко всему прочему в такой структуре ЦП микрооперации фиксирован ной длины обрабатывать проще, чем сложные команды класса x86, имеющие формат переменной длины (рис. 4.9) .

Увеличение тактовой частоты Pentium Pro достигается за счет глубокой конвейеризации. Поскольку блок резервирования обладает высокой гибкостью, можно считать, что конвейер не имеет фиксированного числа ступеней, но минимальное число тактов, в течение которых может быть вы полнена команда, составляет 12. Такие фазы, как обращение к кэшу и декодирование команды, требуют 2,5 такта каждая.

Структурная схема микропроцессора Pentium Pro



БЗД – блок записи данных
БФАЗД – блок формирования записи данных
БФАЗ – блок формирования адреса загрузки
ФТ – фиксированная точка
ПТ – плавающая точка
А - арифметика

Рис. 4.9.

Для достижения максимальной производительности Pentium Pro разработчики разместили кэш-память второго уровня на отдельном кристалле, который монтируется в один корпус с чипом центрального процессора. Благодаря тому, что соединения между ними не выходят за пределы корпуса (т. е. имеют минимальную протяженность) и выполнены по схеме точка - точка, инженеры компании смогли использовать в интерфейсе типа процессор - кэш нестандартные уровни сигналов и получить высокую пропускную способность тракта. Чип кэша второго уровня выпускается в двух модификациях: емкостью 256 и 512 Кбайт - и обеспечивает передачу 8 байт за такт даже при рабочей частоте процессора 200 МГц.

Для примера рассмотрим микроархитектуру процессора Pentium II. Основная отличительная черта процессоров семейства P6 — использование алгоритмов "динамического выполнения команд" (dynamic execution), которые построены на основе трех базовых концепций: предсказании переходов (branch prediction), динамическом анализе потока данных (dynamic data flow analysis) и спекулятивном выполнении инструкций (speculative execution).

Предсказание переходов — это концепция, которая реализована не только в микроархитектуре процессоров семейства P6, но и в микроархитектуре ряда других высокопроизводительных процессоров (например, процессоров мэйнфреймов). Суть ее заключается в следующем. На вход процессора поступает поток инструкций для их последующего исполнения. Инструкции поступают в том порядке, в котором они содержатся в коде программы, исполняемой в данный момент процессором. Как только на входе процессора появляется очередная порция инструкций для исполнения, ее содержимое анализируется с целью найти точки ветвления в исполняемом потоке инструкций и предсказать наиболее вероятные пути (ветви), по которым пойдет обработка инструкций после этих точек ветвления. Инструкции, принадлежащие ветвям с наибольшей вероятностью выполнения, тут же ставятся в очередь на исполнение.

Основная идея всех этих манипуляций заключается в том, чтобы заставить процессор выполнить инструкции, которые принадлежат ветвям с наибольшей вероятностью выполнения, "вне очереди" — то есть не дожидаться того момента, когда очередь на выполнение дойдет до этих ветвей естественным образом (согласно порядку поступления инструкций на вход процессора и, соответственно, контексту выполняемой программы), а загрузить эти ветви на выполнение раньше этого момента. Таким образом обеспечивается более полная загрузка и, соответственно, более высокая производительность процессора.

Конечно, такое преждевременное исполнение инструкций может оправдать себя только в том случае, если алгоритм нахождения наиболее вероятных ветвей работает достаточно хорошо. Действительно, если ветвь угадана неверно, то процессору придется исполнить инструкции, принадлежащие как неверно угаданной, так и альтернативной ветви, проделав

тем самым двойную работу. Если бы такая ситуация наблюдалась часто, то использование этой методики было бы невыгодно. Судя по тому, что концепция "предсказания переходов" активно используется производителями процессоров, соответствующая алгоритмика развита достаточно хорошо. В процессоре Pentium II за предсказание переходов "отвечает" Fetch/Decode Unit (модуль загрузки/декодирования инструкций).

Динамический анализ потока данных включает в себя выполняемый в режиме реального времени анализ зависимости инструкций от исходных данных и значений регистров процессора, а также определение возможности исполнения и непосредственное исполнение инструкций в порядке, отличном от порядка их первоначальной постановки в очередь на исполнение, (out-of-ordere execution).

Dispatch/Execute Unit (модуль диспетчеризации/исполнения инструкций) процессора Pentium II может одновременно следить за ходом исполнения множества инструкций и выполнять их в таком порядке, который позволяет оптимизировать загрузку вычислительных ресурсов процессора. В это же самое время Dispatch/Execute Unit следит за целостностью данных, над которыми проводятся вычисления.

Выполнение инструкций в порядке, отличном от порядка их постановки в очередь на исполнение (out-of-order execution), позволяет избежать простоя вычислительных ресурсов даже в том случае, когда в L1-кэше нет данных, необходимых для исполнения инструкции, или между инструкциями есть зависимость данных, и зависимая инструкция не может быть исполнена. Например, в результате исполнения инструкции "A" получаются данные, которые используются при исполнении инструкции "B"; соответственно, инструкция "B" не может быть исполнена раньше, чем инструкция "A".

Спекулятивное выполнение инструкций — это способность процессора исполнить инструкции в порядке, отличном (как правило, с опережением) от порядка во входном потоке инструкций (что определяется кодом исполняемой программы), но завершить и возвратить (commit) результаты исполнения инструкций в порядке, соответствующем оригинальному входному потоку инструкций.

В процессоре Pentium II спекулятивное выполнение инструкций возможно благодаря тому, что этап "диспетчеризации и выполнения инструкций" (dispatching and executing of instructions) отделен от этапа "завершения и возвращения результатов" (commitment of results).

Используя динамический анализ потока данных, Dispatch/Execute Unit процессора исполняет все инструкции, находящиеся в пуле инструкций (instruction pool) и готовые к исполнению, после чего записывает результаты их исполнения во временные регистры.

В это время Retire Unit (модуль завершения и удаления инструкций) последовательно просматривает пул инструкций и ищет исполненные инструкции, которые не имеют зависящих от них других инструкций,

следовательно, могут считаться исполненными и готовыми к извлечению из пула инструкций. Найденные инструкции извлекаются из пула инструкций, в том порядке, в каком они поступили в очередь на исполнение, результаты их исполнения возвращаются (commited), записываются в оперативную память и/или в IA-регистры (Intel Architecture registers — регистры общего назначения процессора) и регистры данных математического сопроцессора (FPU — floating-point unit)), после чего инструкции удаляются из пула инструкций.

Алгоритмика динамического выполнения команд, включающая предсказание переходов, динамический анализ потока данных и спекулятивное выполнение инструкций, снимает ограничения традиционного "линейного" подхода, при котором весь цикл исполнения состоял из двух этапов — загрузки и выполнения инструкций, а сами инструкции обрабатывались в том порядке, в котором они поступали в очередь на исполнение.

Ядро и подсистемы памяти Pentium

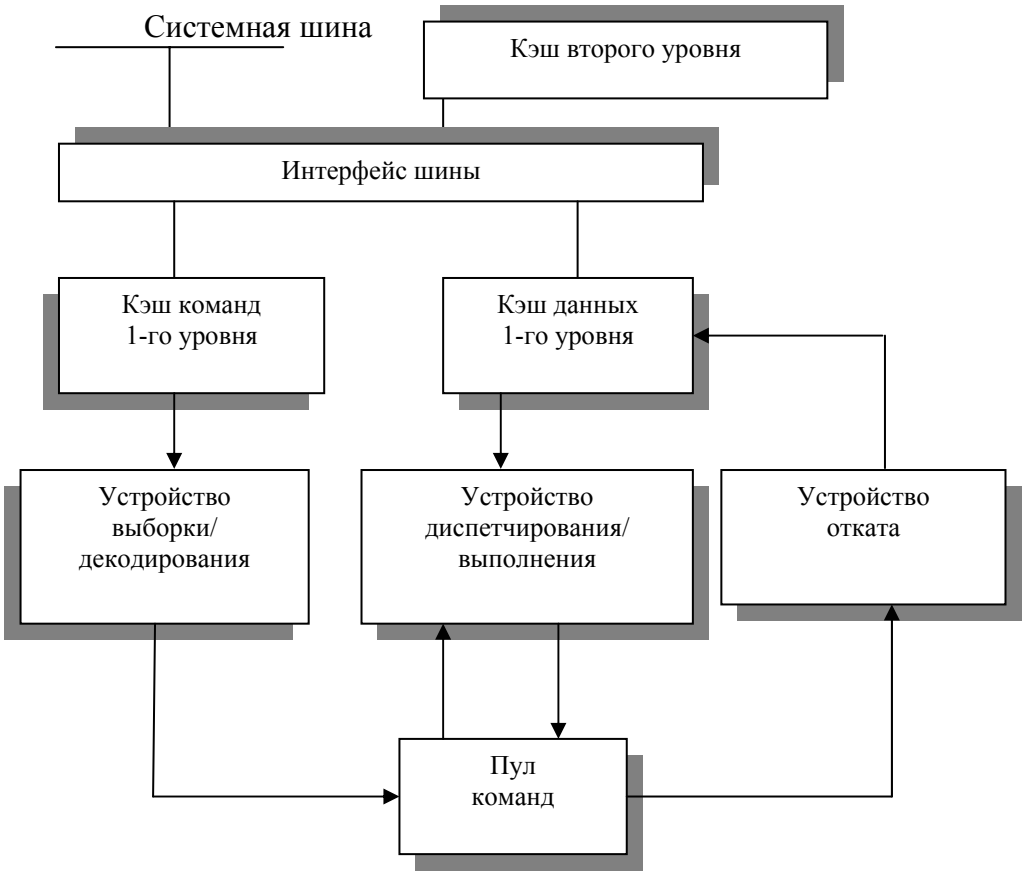


Рис.4.10.

Процессор Pentium II построен на основе семи базовых модулей (рис. 4.10) — Fetch/Decode Unit (модуль загрузки/декодирования инструкций), Dispatch/Execute Unit (модуль диспетчеризации/исполнения инструкций), Retire Unit (модуль завершения и удаления инструкций). Instruction Pool (пул

инструкций, его также называют Reorder Buffer — буфер переупорядочивания инструкций). Bus Interface Unit (модуль внешнего интерфейса), LI ICache (L1-кэш для инструкций) и LI DCache (L1-кэш для данных).

Fetch/Decode Unit предназначен для приема входного потока инструкций исполняемой программы, поступающего из L1-кэша инструкций, и их последующего декодирования в поток микроопераций (рис. 4.11).

Этот модуль работает следующим образом. Прежде всего, блок Next_IP вычисляет индекс (порядковый номер) инструкции, содержащейся в L1-кэше инструкций, которая должна быть обработана следующей — то есть извлечена из L1-кэша инструкций и передана для последующего декодирования.

Индекс этой инструкции вычисляется блоком Next_IP на основе поступающей в него информации о прерываниях, которые были переданы в процессор для обработки, возможных предсказанных переходах (предсказание выполняется блоком Branch Target Buffer), и сообщениях о неправильно предсказанных переходах (branch-misprediction), которые поступают от целочисленных вычислительных ресурсов, расположенных в модуле Dispatch/Execute Unit. После вычисления индекса следующей обрабатываемой инструкции Li-кэш инструкций извлекает две строки кэшированных данных (cache line) — ту, которая соответствует вычисленному индексу, и следующую за ней, — а затем передает для декодирования извлеченные 16 байт, которые содержат IA-инструкции (Intel Architecture). Начало и конец IA-инструкций маркируются.

Далее маркированный поток байт обрабатывается сразу тремя параллельно работающими декодерами, которые отыскивают в нем IA-инструкции. Каждый декодер преобразует найденную IA-инструкцию в набор триадных микроопераций (uops) — триадных в том смысле, что микрооперация проводится над двумя исходными логическими операндами, а в результате ее выполнения получается только один логический результат. Микрооперация — это примитивная инструкция, которая может быть выполнена одним из вычислительных ресурсов, расположенных в модуле Dispatch/Execute Unit.

Из трех декодеров два — простые, которые могут преобразовывать только IA-инструкций, требующие выполнения одной микрооперации, а третий декодер — более совершенный; он может преобразовывать IA-инструкции, требующие выполнения от одной до четырех микроопераций. Таким образом, за один такт работы процессора все три декодера могут в сумме сгенерировать максимум шесть микроопераций. Для преобразования еще более сложных IA-инструкций используется микрокод, который содержится в блоке Microcode Instruction Sequencer и представляет собой набор предварительно запрограммированных последовательностей обычных микроопераций.

Устройство выборки/декодирования

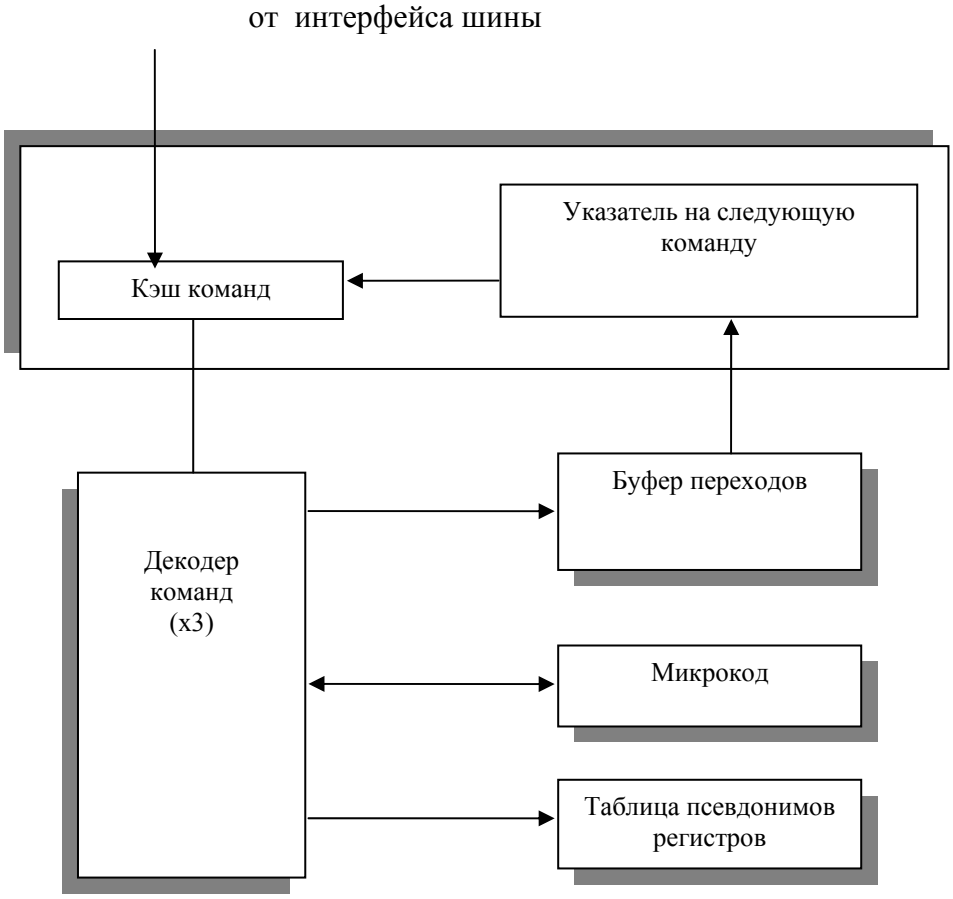


Рис. 4.11.

Полученные таким образом микрооперации передаются в блок Register Alias Table Allocate, где все содержащиеся в микрооперациях адреса lA-регистров преобразуются в адреса внутренних физических регистров процессора семейства P6 — тем самым IA-архитектура и Pб-архитекура оказываются развязанными. Это существенно увеличивает возможности работы процессора при вычислениях, так как, во-первых, отпадает необходимость следить за целостностью содержимого IA-регистров при исполнении инструкций, во-вторых, адресное пространство перестает быть ограниченным возможностями IA-архитектуры и может быть значительно расширено, что приводит к росту скорости вычислений, и, в-третьих, такая переадресация обеспечивает возможность спекулятивного исполнения инструкций — далее все вычисления ведутся во внутренней P6-архитектуре процессора, а IA-архитектура снова появляется "на сцене" только на этапе завершения инструкций в модуле Retire Unit.

На этом же этапе к каждой микрооперации как информационной единице добавляются флаги состояния, в которые записывается информация об ее статусе. После этого микрооперации передаются в пул инструкций.

Instruction Pool (Reorder Buffer). Основное назначение этого модуля — предоставить возможность исполнения микроопераций в произвольном порядке; в том числе, отличном от порядка их генерации.

В тот момент, когда микрооперации попадают в пул инструкций, порядок их следования в потоке соответствует тому порядку, в котором они были сгенерированы в результате декодирования IA-инструкций, поступивших на вход модуля Fetch/Decode Unit, — никакого изменения порядка следования пока не произошло. Пул инструкций представляет собой последовательный массив инструкций; при этом любая из этих инструкций может быть в любой момент времени обработана модулем Dispatch/Execute Unit или Retire Unit — то есть порядок обработки инструкций может быть произвольным и не зависит от первоначального порядка, в котором инструкции поступили в пул. Именно поэтому пул инструкций иногда называют еще буфером переупорядочивания инструкций (Reorder Buffer).

Dispatch/Execute Unit. Этот модуль проверяет состояние микроопераций, содержащихся в пуле инструкций, исполняет их, если есть такая возможность, и записывает полученные результаты обратно в пул инструкций (рис. 4.12).

Reservation Station — основной управляющий блок модуля Dispatch/Execute Unit. Именно он планирует порядок исполнения и занимается диспетчеризацией (распределением между вычислительными ресурсами) микроопераций. Этот блок последовательно просматривает пул инструкций в поисках микроопераций, которые готовы к исполнению — таковыми считаются микрооперации, у которых готовы (т.е. вычислены/загружены) исходные операнды, — и передает (распределяет, диспетчеризует) их на исполнение свободным вычислительными ресурсам, которые могут исполнить микрооперацию. Результаты исполнения микрооперации записываются в пул инструкций и хранятся там вместе с самой микрооперацией до тех пор, пока последняя не будет завершена — этим занимается уже модуль Retire Unit.

Следует подчеркнуть, что жесткого, заранее предопределенного порядка исполнения микроопераций не существует — они исполняются сразу же, как только бывают готовы их операнды и свободен соответствующий вычислительный ресурс. В том случае, если одному и тому же ресурсу может быть одновременно передано на исполнение более одной микрооперации, последние исполняются по принципу псевдо-FIFO (First In First Out) — первой исполняется та микрооперация, которая раньше попала в пул инструкций.

Reservation Station имеет пять портов, через которые организуется обмен данными с пятью вычислительными ресурсами. Поэтому Dispatch/Execute Unit может за один такт исполнить максимум пять микроопераций. Однако при реальной работе с постоянной равномерной нагрузкой на процессор наиболее типична ситуация, когда за один такт исполняется три микрооперации.

Структура устройства диспетчирования/выполнения

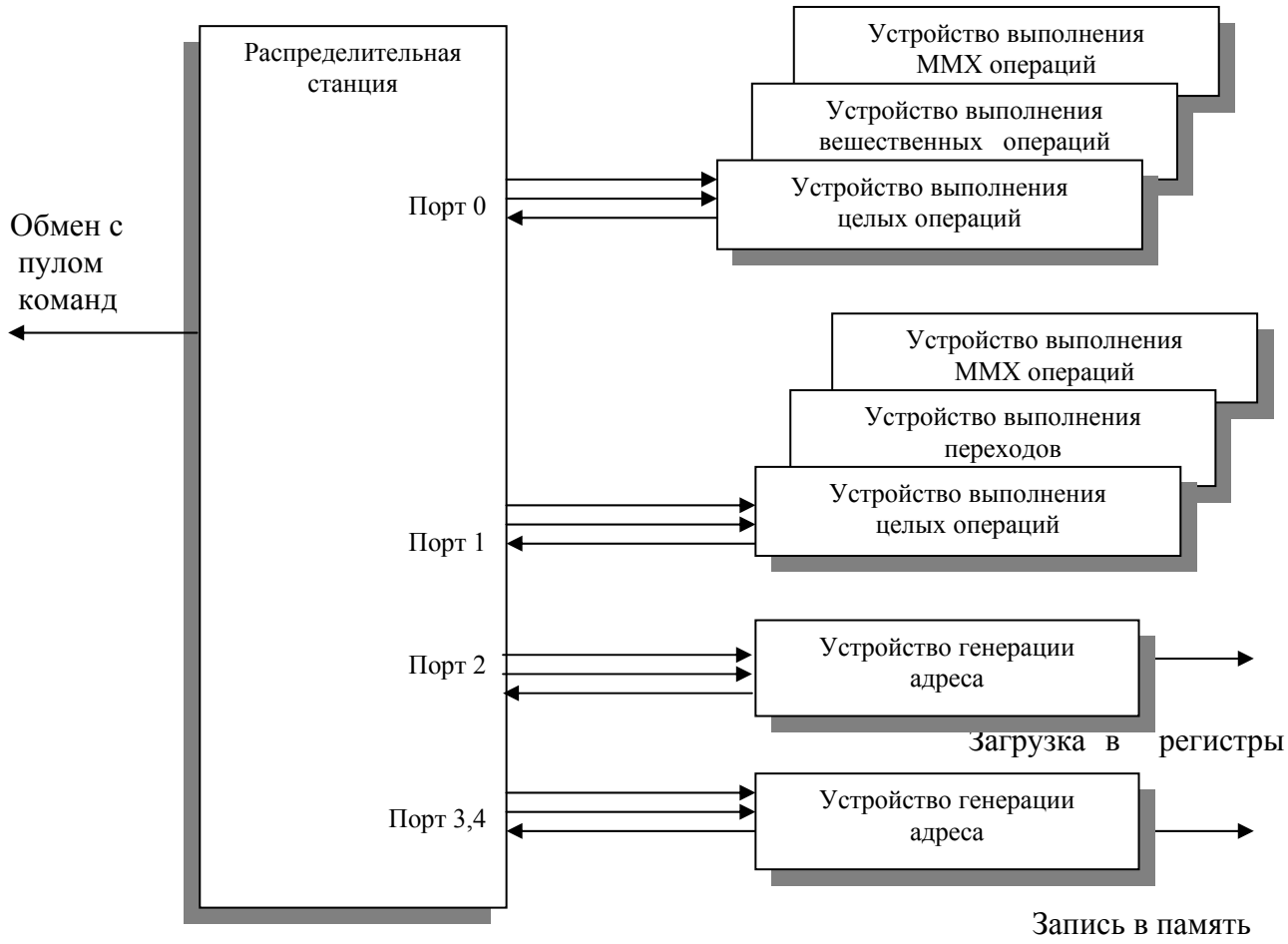


Рис. 4.13.

Retire Unit — модуль (рис. 4.14), который знает как и когда завершить (commit) временные внутренние спекулятивные вычисления, выполненные в P6-архитектуре, преобразовать их и вернуть окончательный результат в IA-архитектуре.

Retire Unit постоянно сканирует содержимое пула инструкций и проверяет статус хранящихся в нем микроопераций. Как только находится исполненная и готовая к удалению из пула микрооперация. Retire Unit преобразует результаты ее исполнения, хранящиеся во внутреннем представлении процессора (то есть во внутренних регистрах, в контексте P6-архитектуры), к представлению в IA-архитектуре и записывает результат исполнения в оперативную память и/или в IA-регистры. После этого микрооперация удаляется из пула инструкций.

Устройство отката

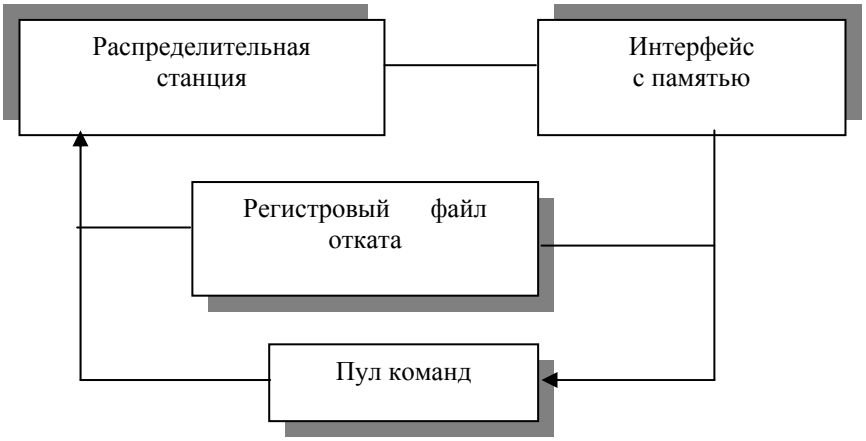


Рис. 4.14.

Тонкость этой процедуры заключается в том, что результаты исполнения микроопераций должны быть возвращены в контексте IA-архитектуры в том же порядке, в каком эти микрооперации были сгенерированы в модуле Fetch/Decode Unit при декодировании входного потока инструкций исполняемой программы.

Ситуация усложняется еще тем, что все это происходит на фоне непрекращающегося потока всевозможных прерываний, точек останова, ошибок предсказания переходов, а также внештатных ситуаций в работе процессора, которые нужно успевать обрабатывать.

Retire Unit процессора Pentium II способен завершить и удалить до трех микроопераций за один такт работы процессора.

Интерфейс с шиной

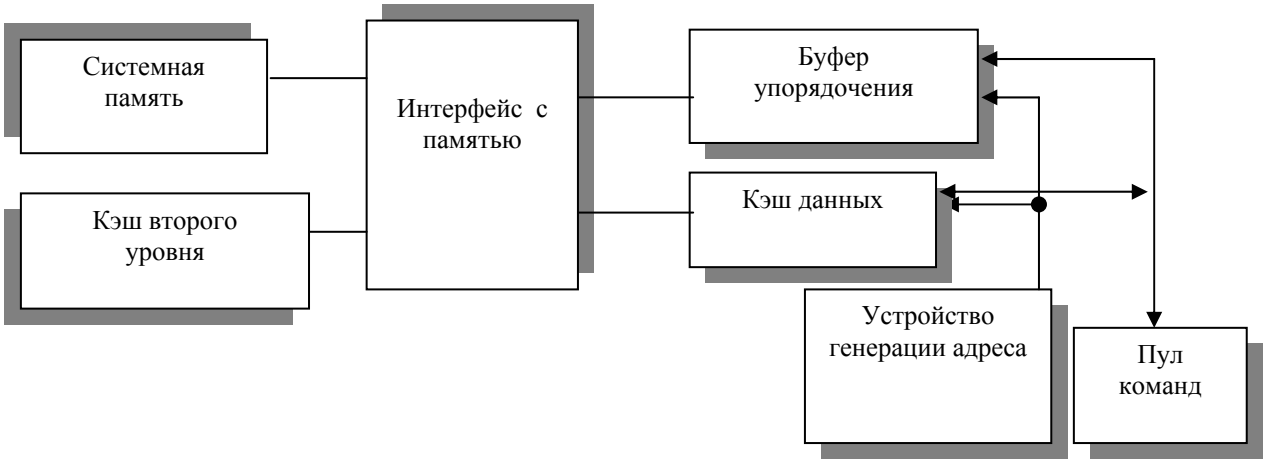


Рис. 4.15.

Bus Interface Unit. Этот модуль (рис. 4.15) отвечает за обмен данными между Ll-кэшом инструкций, L1-кэшом данных, системной шиной и L2-кэшом.

При чтении из памяти должны быть заданы адрес памяти, размер блока считываемых данных и регистр-назначение. Команда чтения кодируется одной микрокомандой.

При записи надо задать адрес памяти, размер блока записываемых данных и сами данные. Поэтому команда записи кодируется двумя микрокомандами: первая генерирует адрес, вторая готовит данные. Эти микрокоманды планируются независимо и могут выполняться параллельно; они могут переупорядочиваться в буфере записи.

Запись в память никогда не выполняется опережающим образом, так как нет эффективного способа организации отката в случае неверного предсказания. Разные команды записи никогда не переупорядочиваются друг относительно друга. Буфер записи инициирует запись, только когда сформированы и адрес, и данные, и нет ожидающих выполнения более ранних команд записи.

При изучении вопроса о возможности и целесообразности переупорядочения доступа к памяти инженеры "Intel" пришли к следующим выводам.

Команда записи не должна обгонять идущую впереди команду записи, так как это может лишь незначительно увеличить производительность.

Можно запретить командам записи обгонять команды чтения из памяти, так как это приведет лишь к незначительной потере производительности.

Запрет командам чтения обгонять другие команды чтения или команды записи может повлечь существенные потери в производительности.

Поэтому была реализована архитектура подсистемы памяти, позволяющая командам чтения опережать команды записи и другие команды чтения.

Буфер упорядочения памяти служит в качестве распределительной станции и буфера переупорядочивания. В нем хранятся отложенные команды чтения и записи, и он осуществляет их повторное диспетчирование, когда блокирующее условие (зависимость по данным или недоступность ресурсов) исчезает.

4.5. Микроархитектура процессоров семейства AMD

В процессоре AMD-K6-2 реализована так называемая "Enhanced RISC86"-микроархитектура. Напомним, что RISC — это аббревиатура от Reduced Instruction Set Computing ("вычисления с сокращенным набором команд"). RISC-процессор обладает меньшим числом команд фиксированной длины. Упрощенная структура позволяет RISC-процессору развивать более высокую скорость. Типичные представители RISC-процессоров — Alpha от DEC, SPARC от SUN, PowerPC от IBM.

В противоположность этому CISC — сокращение от Complex Instruction Set Computing ("вычисления со сложным набором команд"). Все члены семейства x86 — типичные представители CISC-процессоров со сложными, но

удобными наборами команд. Что касается AMD-K6-2, то речь в данном случае идет об объединенной архитектуре на основе преобразования x86-команд в более простые в обращении RISC-инструкции,

Основная ее особенность состоит в том, что внешние x86-инструкции, поступающие на обработку в процессор, преобразуются во внутренние RISC86-инструкции, которые и исполняются процессором. Вместо того чтобы напрямую исполнять сложные x86-инструкции с переменной длиной от 1 до 15 байт, процессор обрабатывает поток простых RISC86-инструкций фиксированной длины.

В состав процессора AMD-K6-2 (рис. 4.16) входят несколько основных модулей; L1-кэш данных (Level-One Dual Port Data Cache), Li-кэш инструкций (Level-One Instruction Cache) с кэшем предварительного декодирования (Predecode Cache), модуль декодирования (Multiple Instruction Decoders), центральный планировщик (Centralized RISC86 Operation Scheduler), вычислительные блоки (Execution Units) и модуль предсказания переходов (Branch Logic).

L1-кэш инструкций и данных, предварительное декодирование. L1-кэш состоит из двух независимых блоков: L1-кэша данных (Level-One Dual Port Data Cache) и L1-кэша инструкций (Level-One Instruction Cache) с кэшем предварительного декодирования (Predecode Cache). L1-кэш данных предназначен только для хранения данных и имеет объем 32 Кбайт- Несколько сложнее обстоит дело с L1-кэшем инструкций: наряду с инструкциями, для хранения которых предназначены 32 Кбайт памяти, в нем хранятся так называемые "биты преддекодирования" (predecode bits) — для них отведено 20 Кбайт памяти. Дело в том, что после загрузки инструкции в L1-кэш инструкций выполняется ее предварительное декодирование (predecoding) — к каждому байту инструкции добавляется пять бит (из этого и следует соотношение 32 Кбайт/20 Кбайт = 8/5), в которые записывается информация о количестве байт, оставшихся до начала следующей инструкции. Эта информация используется на этапе декодирования x86-инструкций в RISC8б-инструкции. После того, как L1 -кэш инструкций полностью заполнится данными, инструкции вместе с преддекодированными битами передаются в буфер инструкций (Instruction Buffer).

Модуль декодирования (Multiple Instruction Decoders), Модуль декодирования извлекает x86-инструкций (до 16 байт данных с инструкциями за один такт) с битами преддекодирования из буфера инструкций (Instruction Buffer), определяет границы инструкций и преобразует их в RISC86-инструкции. Непосредственно преобразованием занимаются четыре декодера; два для декодирования простых (Short Decoder #1, Short Decoder #2) и два для декодирования сложных x86-инструкций *(Long* Decoder, Vector Decoder). Одновременно могут работать либо два декодера Short Decoder #1 и Short Decoder #2, либо декодер Long Decoder, либо декодер Vector Decoder.

Микроархитектура процессора AMD-K6-2

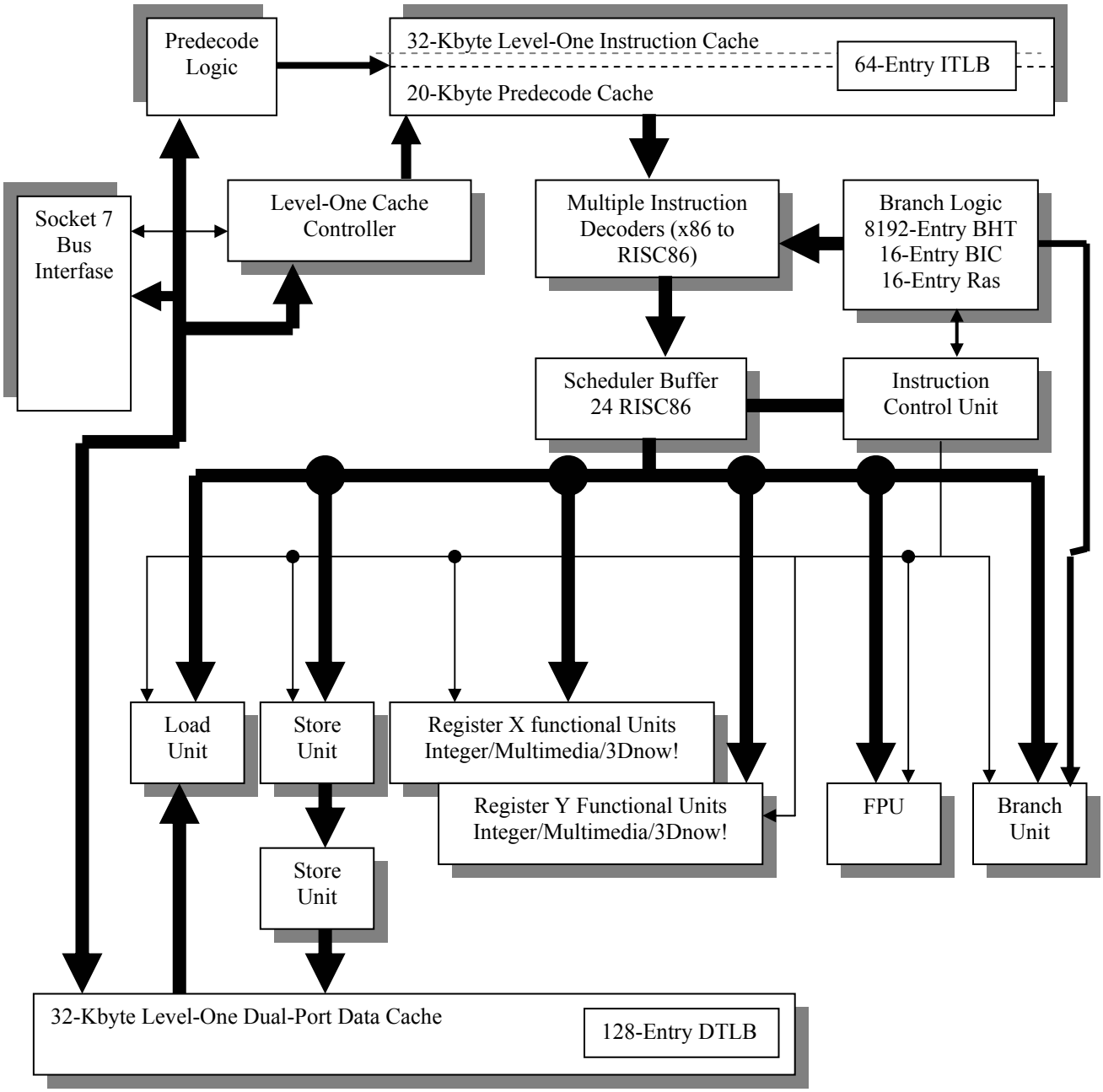


Рис. 4.16.

Два декодера Short Decoder *#1* и Short Decoder #2 работают параллельно и обрабатывают наиболее часто используемые x86-инструкций — move, shift, branch, ALU, FPU, а также инструкции из наборов команд MMX и 3DNow! Декодеры Short Decoder #1 и Short Decoder *#2* обрабатывают только часто используемые (most commonly-used) x86-инструкций длиной не более семи байт- Каждый может преобразовать только одну такую x86-инструкцию и сгенерировать 0 (например, при обработке x86-инструкций NOP), одну или две

RISC86-инструкции за такт. Таким образом, за один такт оба декодера могут сгенерировать до 4 PISC86-инструкций.

Редко используемые инструкции (semi-commonly-used) длиной до семи байт и обычные инструкции (commonly-used) с длиной большей семи байт, но меньшей или равной 11 байтам обрабатываются декодером Long Decoder, который может декодировать только одну такую x86-инструкцию и сгенерировать до 4 RISC86-инструкций за такт. Все остальные преобразования (более сложные инструкции, прерывания, и. т. д.) выполняются декодером Vector Decoder. В этом случае Vector Decoder генерирует набор первых RISC86-инструкции и адрес заранее предопределенного набора последующих инструкций, который хранится в ROM-памяти (On-Chip ROM) и извлекается блоком RISC86 Sequencer.

Все наборы RISC86-операций, генерируемые декодерами и извлекаемые из On-Chip ROM всегда (!) состоят из групп, содержащих по четыре RISC86-операции. В том случае, если их получилось меньше, недостающее количество заполняется пустыми RISC86-инструкциями NOP. Например, если Long Decoder преобразовал x86-инструкцию в три RISC86-инструкции, то к ней добавляется одна RISC86-инструкция NOP. Получившийся лоток из таких групп поступает в буфер планировщика (Scheduler Buffer) — за один такт всегда передается группа из четырех RISC-операций. Центральный планировщик (Centralized RISC86 Operation Scheduler). Планировщик — это сердце процессора AMD-K6-2. Он следит за процессом исполнения RISK86-инструкций, приведением результата их исполнения к x86-архитектуре, а также возвращением результатов спекулятивного выполнения x86-инструкций в соответствии с их порядком поступления на вход процессора.

В буфере планировщика может одновременно содержаться до 24 RISC86-инструкций. Любая из них может быть в любой момент передана на исполнение соответствующему вычислительному блоку (store, load, branch, register X integer/multimedia, register Y integer/multimedia, floating-point), если, конечно, последний свободен. Таким образом, реализуется исполнение инструкций в порядке, отличном от порядка их поступления в буфер (out-of-order execution). В общей сложности планировщик может передать на выполнение шесть и завершить (retire) также шесть RISC86-инструкций за такт, Вычислительные блоки (Execution Units). Процессор AMD-K6-2 содержит 30 параллельных вычислительных блоков — Store Unit, Load Unit, Integer X ALU, Integer Y ALU, MMX ALU (X), MMX ALU (Y), MMX/3DNow! Multiplier, 3DNow! ALU, FPU и Branch Unit. Каждый блок работает независимо от остальных, так что несколько блоков могут обрабатывать переданные им на исполнение RISC86-инструкции.

Микроархитектура процессора AMD Athlon

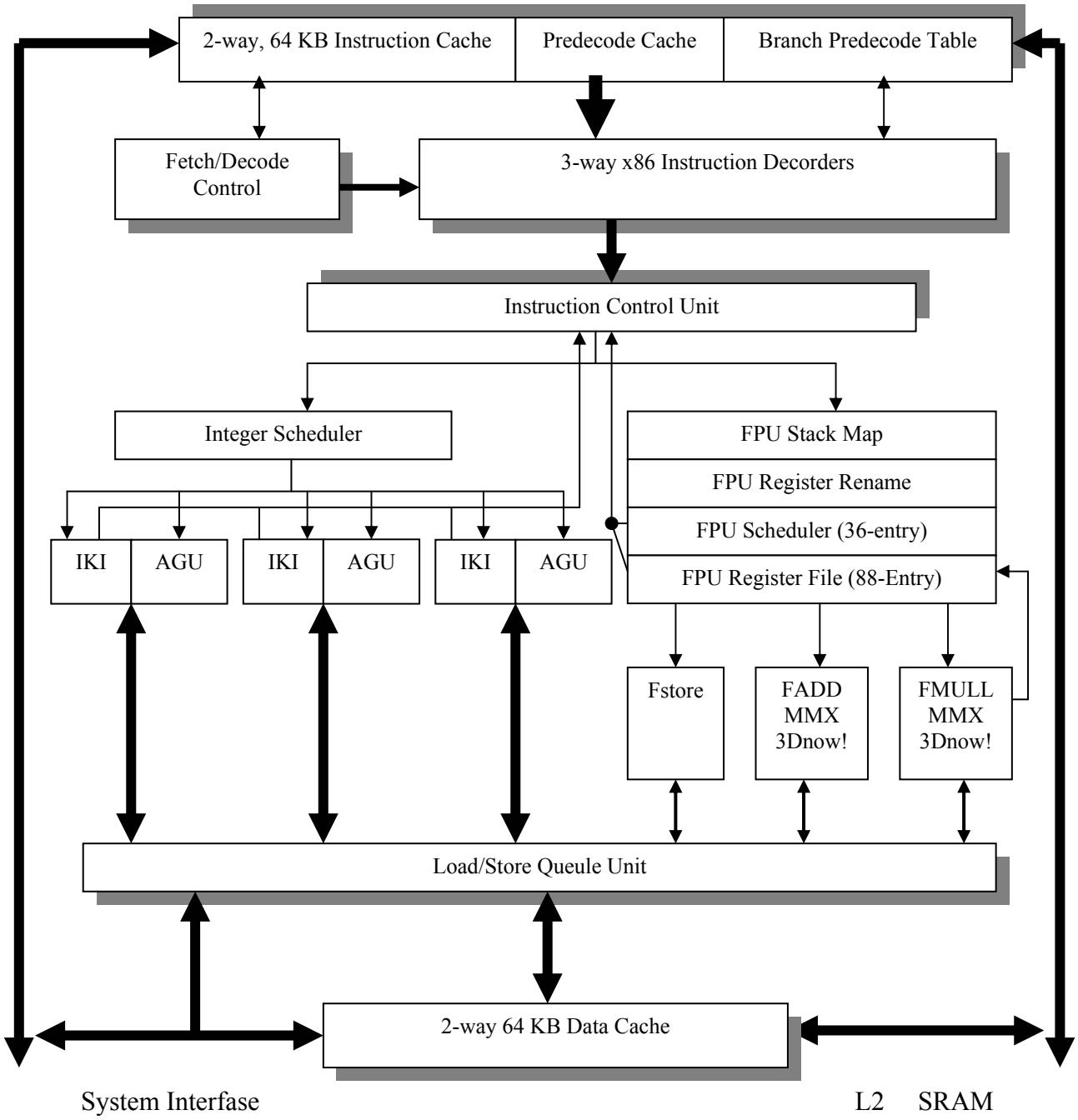


Рис. 4.17.

Integer, MMX- и 3DNow! - инструкции передаются по двум независимым шинам — Register X Issue Bus и Register Y Issue Bus. При этом блоки Integer X ALU и MMX ALU (X) подключены только к шине Register X Issue Bus, а Integer Y ALU и MMX ALU (У) — только *к* шине Register Y Issue Bus. А вот блоки MMX/3DNow! Multiplier и 3DNow! ALU подключены сразу к обеим шинам, как и блок MMX Shifter, функция которого заключается в том, чтобы переключать блоки MMX/3DNow! Multiplier и 3DNow! ALU между шинами.

Модуль предсказания переходов (Branch Logic). Назначение этого модуля, как следует из его названия, состоит в предсказании возможных переходов.

Во всех "старых процессорах AMD, Модуль вычислений с плавающей точкой был неконвейерным, что не позволяло начать выполнять новую команду пока не закончиться выполнение предыдущей. Это приводит к сильному падению производительности всей системы. До сих пор разработчики AMD не вносили никаких изменений в FPU, рассчитывая на свой блок 3Dnow!

в Athlone AMD (рис. 4.17) впервые представляет новый, полностью конвейерный FPU модуль, позволяюший выполнять до трех операций за такт.

Обратите внимание на три вычислительных блока и на то, как модули Stack Map, Registry Rename, Scheduler с 36 входами и FPU Register File с 88 входами позволяют разделить вычисления между ними (рис. 4.18).

Рабочая схема модуля вычислений с плавающей точкой

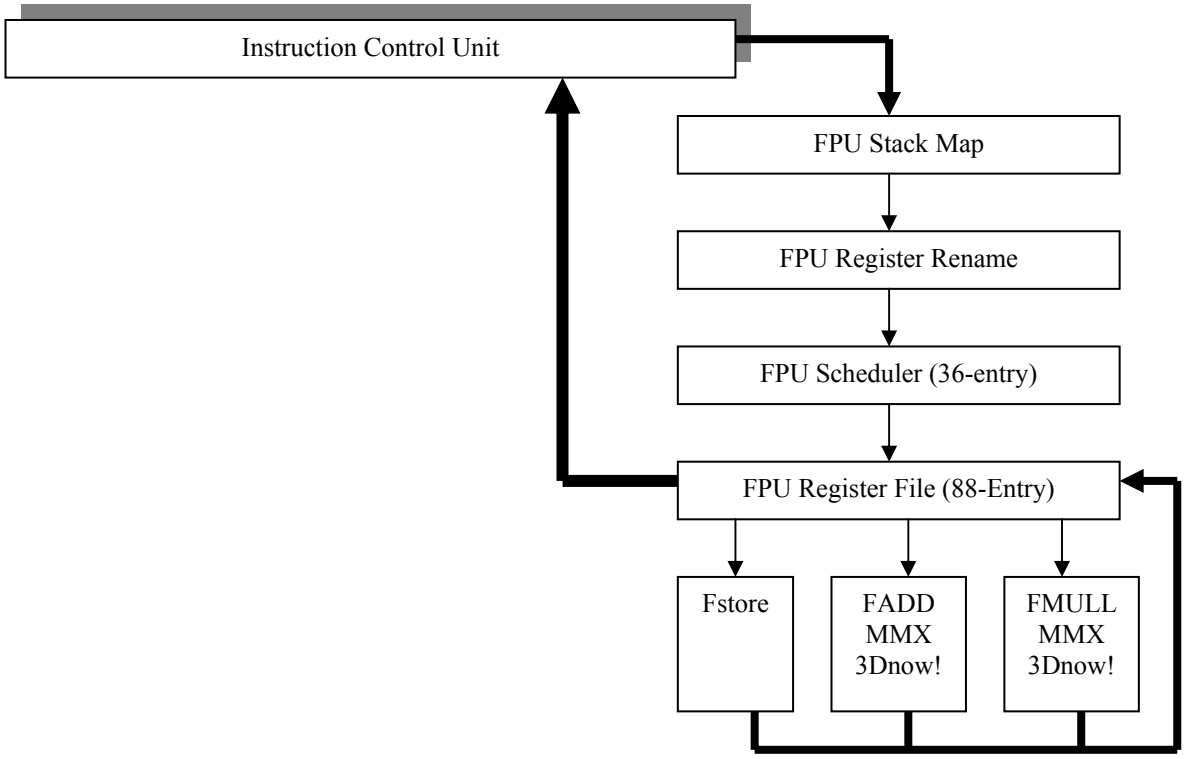


Рис. 4.18.

FPU в процессорах Pentium III и Celeron разделен на два модуля FADD и FMUL; первый, полностью конвейерный, выполняет простые вычисления, в то время как второй выполняет более сложные вычисления и не полностью конвейерный. Естественно, что наличие трех, полностью конвейерных модулей, вместо двух, из которых только один

полностью конвейерный, позволяет получить лучшую производительность в приложениях, активно использующих вычисления вещественных чисел.

По сравнению с FPU, целочисленные модули AMD всегда были достаточно производительными, но, несмотря на это, в Athlon произошли некоторые изменения.

В новом блоке целочисленных вычислений используется три конвейерных модуля, способных выполнять три операции одновременно. Глубина этих конвейерных модулей составляет 10 шагов, что, по словам AMD, является самым оптимальным.

Блок 3DNow! также претерпел ряд изменений. 19 новых SIMD инструкций были добавлены к оригинальному 3Dnow!, чтобы увеличить производительность целочисленных и вещественных операций. Также добавлены пять новых DSP инструкций предназначенных для использования в мультимедиа приложениях (MP3, AC3, Mpeg2 encoding и decoding). Используя расширенный набор 3Dnow! Инструкций, можно получить до 20% прироста производительности по сравнению с стандартным набором и около 50% по сравнению с приложениями, не использующими 3Dnow!

Одной из интересных особенностей нового процессора является применение новой архитектуры системной шины — EV6. Такая же архитектура применяется в системах на процессорах Alpha. Применение EV6 связано с желанием AMD использовать Athlon в производительных многопроцессорных системах.

Сравнение EV6 и GTL+ (Pentium III и Celeron) показывает, что первая архитектура использует технологию Point to Point, в которой каждый процессор имеет свою отдельную часть шины, в то время, как в GTL+ процессоры должны делить одну "широкую" шину. Это объясняет, почему многопроцессорные системы на GTL+ шине очень трудно найти. EV6 поддерживает до 14 процессоров, но пока самые оптимальные теоретические расчеты предлагают использовать не более восьми процессоров.

Организация системной шины

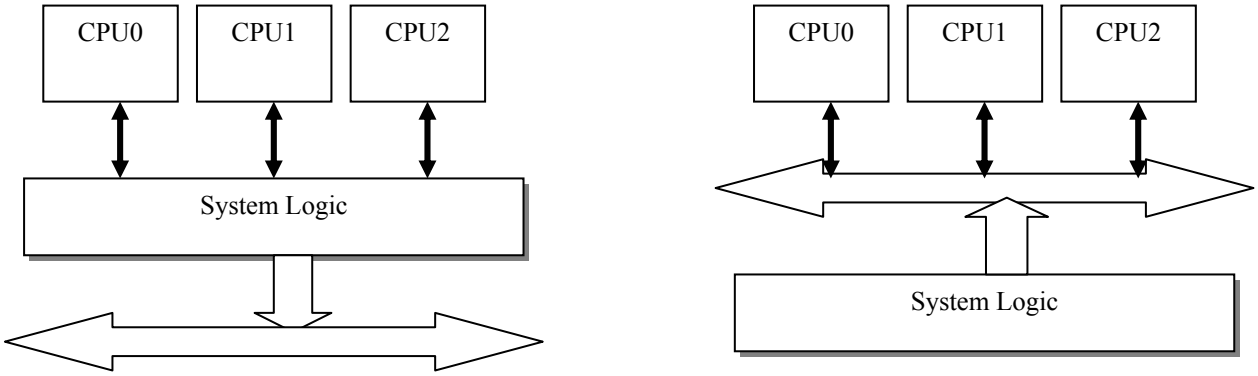


Рис. 4.19.

Левая блок-схема показывает организацию EV6. Каждый процессор имеет собственную шину для соединения с логическими схемами, даже если в системе установлено несколько процессоров. На правой блок-схеме Вы можете видеть решение, основанное на GTL+ шине: ширина системной шины зафиксирована и делится между доступными ей процессорами (рис. 4.19).

5.Принципы организации системы прерывания программ

5.1. Классы сигналов прерывания

Для обеспечения перехода от одной программы к другой в мультипрограммной ЭВМ вводится так называемый режим прерывания программ.

Прерывание программы — способность процессора прекращать выполнение текущей программы и ее управление при возникновении определенных условий. Сигналы, вызывающие прерывание программы, называются сигналами прерывания или запросами прерывания.

В зависимости от условий возникновения сигналы прерывания подразделяют на пять классов:

1. Прерывание от схем контроля, или машинное прерывание, возникает в случае обнаружения ошибок в каких-либо блоках и устройствах ЭВМ. При этом происходит переключение к диагностической программе, позволяющей локализовать место неисправности.

2. Программное прерывание, или прерывание из-за ошибок программы, возникает при обнаружении ошибок в программе или при появлении необычных ситуаций при ее выполнении. Например, при переполнении разрядной сетки, делении на нуль и т. п.

Внешнее прерывание возникает в случае появления сигналов от внешних объектов: датчиков времени (электронных часов), кнопок запросов на пультах инженера и оператора и других ЭВМ, подключенных к данной ЭВМ; аппаратуры передачи данных по линиям связи, от датчиков технологических процессов и т. п.

3. Прерывание от устройств ввода-вывода позволяет процессору получать информацию о состоянии каналов и периферийных устройств и отвечать на эти сигналы. Это прерывание от устройств ввода-вывода сформируется, когда канал и ПфУ не могут выполнить за данную операцию; при возникновении особой ситуации в процессе выполнения операции ввода-вывода (ошибка в информации, обрыв перфоленты и бумаги, замятие перфокарты и др.); в момент окончания операции ввода-вывода.

4. Прерывание при обращении к управляющим подпрограммам-диспетчеру в случаях, когда предусмотрено выполнение каких-либо действии

по управлению ЭВМ и ВС. Например, если при выполнении рабочей программы нужно выполнить действие, выходящее за пределы возможностей системы команд ЭВМ, то процессор формирует запрос на прерывание в виде обращения к соответствующей подпрограмме, которая реализует затребованное действие, после чего управление вновь передается прерванной программе. Путем обращения к управляющим программам организуется ввод-вывод, перераспределение памяти, прекращается выполнение программы.

5.2. Распределение прерываний в ПК на базе процессоров x86

Все ПК, базирующиеся на процессорах семейства x86 управляются с помощью прерываний, генерируемых аппаратным или программным обеспечением. Эти прерывания могут быть разделены на семь категорий:

- прерывания микропроцессора;
- аппаратные;
- программные;
- операционной системы;
- BASIC;
- адресное и общего назначения.

Прерывания микропроцессора и аппаратные прерывания встроены в процессор и аппаратное обеспечение ПК. Восемь из 16 возможных прерываний не могут быть изменены. Прерывания распределены следующим образом, см. табл. 5.1.

Таблица 5.1.

Распределение аппаратных прерываний

Номер прерывания	Назначение
0	Системный таймер
1	Контролер клавиатуры
2	Сигнал возврата по кадру (видео), связан с 9
3	COM2/COM4
4	COM1/COM3
5	Свободен
6	Контролер FDD
7	LPT1
8	Часы реального времени с автономным питанием
9	Параллельна с 2
10	Свободен
11	Свободен
12	Контролер мыши PS/2
13	Математический сопроцессор
14	Контролер IDE HDD (первый канал)
15	Контролер IDE HDD (второй канал)

Программные прерывания являются часть программ ROM-BIOS. Сами подпрограммы BIOS, вызываемыми этими прерываниями, не могут быть изменены, однако векторы, указывающие на эти подпрограммы, могут быть изменены таким образом, что они будут указывать совсем на другие подпрограммы. Зарезервированные коды - 5, 16-28, 72.

Прерывания операционной системы (ОС) используются самой ОС. Многие программы и языки программирования для выполнения своих основных операций, особенно тех, которые связаны дисковым вводом-выводом, через прерывания ОС используют служебные функции ОС. Зарезервированные коды - 32-255 (используются 32-96).

Прерывания BASIC присваиваются самим BASIC и доступны тогда, когда используется BASIC. Зарезервированные коды - 128-240.

Адресные прерывания являются частью таблицы векторов прерываний и используются для хранения сегментных адресов. С этими прерываниями не связаны какие-либо действительные прерывания или подпрограммы обработки прерываний. Три из них связаны с тремя очень важными таблицами: таблицей инициализации изображения, основной таблицей диска и таблицей графических символов. Эти таблицы содержат параметры, используемые ROM-BIOS при процедурах начального запуска и при генерации графических символов. Зарезервированные коды - 29-31, 68,73.

Прерывания общего назначения устанавливаются нашими программами для временного использования. Зарезервированные коды - 96-106.

5.3. Приоритеты прерываний

Программы, выполнявшиеся до появления запросов прерывания, называют прерываемыми программами. Программы, за требованные запросами прерывания,— прерывающими программами.

На рис. 5.1 приведена временная диаграмма прерывания текущей программы П1. При поступлении запроса прерывания (ЗП) процессор переходит к прерывающей программе П2 не сразу, так как требуется время на анализ системы на запрос прерывания. После перехода к прерывающей программе П2 в течение tз происходит запоминание состояния прерванной программы П1, а затем в течение tп исполняется прерывающая программа П2. Сразу после ее окончания в течение tв восстанавливается состояние прерванной программы П1 и ей передается управление. Сумма времени, затрачиваемого на запоминание состояния прерванной программы П1 и на возврат к ней, называют временем обслуживания программы. Время реакции tр и обслуживания tо=tз+tв характеризуют потери машинного времени на организацию процесса прерывания и быстродействия ЭВМ по обслуживанию запросов прерывания.

Временная диаграмма возможного процесса прерывания программы

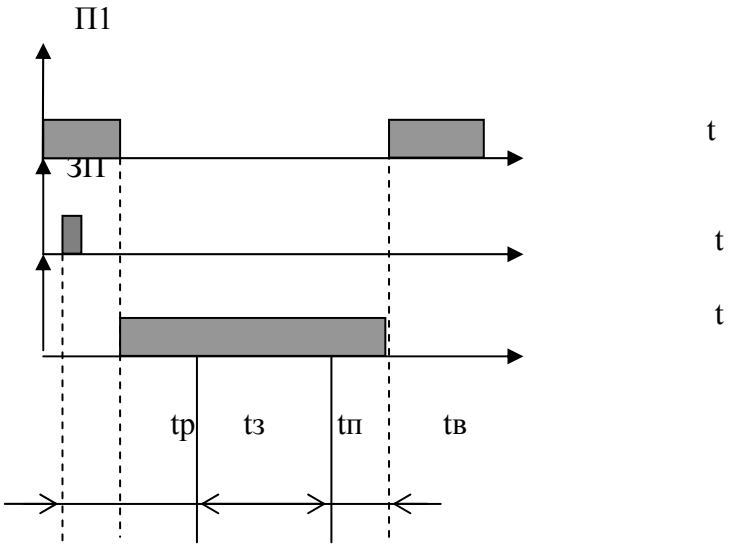


Рис. 5.1.

Максимальное количество программ, прерывающих друг друга вновь возникающими запросами, называют глубиной прерывания. Степень важности запросов на прерывания в общем случае зависит от времени их поступления в систему прерывания программ, характера источников запросов. Поэтому каждому источнику запросов на прерывания присваивается постоянный, как правило, уровень приоритетности, или глубина прерывании. Наивысшим приоритетом (нулевой уровень) прерывания обладают прерывания от схем контроля ЭВМ. Прерывания нулевого уровня могут прервать любую из программ, отвечающих, уровням 1, 2, ... , n при выделении (n+1) уровней.

Первый уровень присваивается прерываниям от устройств ввода-вывода, второй уровень — внешним прерываниям, третий уровень — программным прерываниям и прерываниям при обращении к управляющей программе-диспетчеру. Эти два класса прерываний исключают взаимно друг друга, а поэтому имеют одинаковый приоритет.

Прерывания первого уровня могут прерывать любую из программ второго и третьего уровней приоритетности, но не могут прервать программу с нулевым уровнем приоритетности. На временной диаграмме (рис. 5.2) показано обслуживание запросов ЗПi с учетом их приоритетности.

Определение исполнение программ в системе с учетом приоритетности запросов прерывания

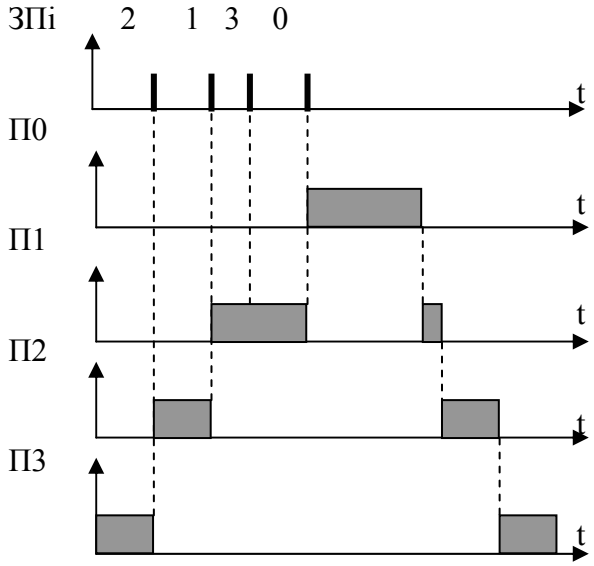


Рис. 5.2.

В связи с введением приоритетов время реакции tp на отдельные запросы увеличивается.

5.4. Защита от прерывания

Процессор должен обеспечивать такой подход к запросам на прерывание, при котором прерывание по отдельным причинам может быть запрещено в течение некоторого промежутка времени, в то время как для других запросов прерывание разрешается. В современных ЭВМ наибольшее распространение получило программное управление приоритетом на основе маски-кода защиты от прерываний, представляющего собой двоичное число, разряды которого соответствуют отдельным причинам или уровням прерывании. Если разряд маски имеет значение 0, то соответствующая причина прерывания замаскирована и процессор не реагирует на данный запрос на прерывание. Если разряд маски равен 1, то соответствующая причина прерывания не замаскирована и процессор воспринимает данный запрос на прерывание.

С замаскированным запросом в зависимости от причины прерывания поступают двояким образом или он игнорируется, или запоминается с тем, чтобы осуществить затребованные действия, когда запрет будет снят. Например, если прерывание вызвано окончанием операции в периферийном устройстве, то его следует, как правило, запомнить, так как иначе ЭВМ останется неосведомленной о том, что периферийное устройство освободилось. Прерывание, вызванное переполнением разрядной сетки при выполнении арифметической операции, следует в случае его маскирования игнорировать,

так как запоминание этою запроса может привести к искажению результата этой программы.

Реализация прерываний в современных ЭВМ осуществляется аппаратными и программными средствами, совокупность которых получила название системы прерывания. С помощью аппаратных средств обнаруживаются сигналы запроса прерывания, организуется запоминание информации, необходимой для начала функционирования программных средств, а также для передачи управления программе прерываний и восстановления старой программы. С помощью программных средств производится запись в память содержимого большинства регистров и информации о состоянии процессора. Все обслуживание прерываний, включая определение номера и типа устройства, вида ошибки, возлагается полностью на программное обеспечение (ПО). Иногда ПО определяет и метод возвращения к старой (прерванной) программе.

Различают два метода обработки прерываний: с опросом и по вектору.

1. Прерывание с опросом. При помощи аппаратных и программных средств осуществляется опрос каждого периферийного устройства, пока не обнаружится то, которое запрашивает прерывание, после чего осуществляется переход на соответствующую подпрограмму обслуживания прерывания, которая и выполняет затребованные действия. При этом приоритет периферийного устройства определяется его местом в последовательности опроса.

2. Прерывание по вектору. Запрос непосредственно передается на соответствующую подпрограмму, т. е. все периферийные устройства обладают одинаковым приоритетом. Поскольку в данном случае опроса не требуется, время реализации прерывания меньше, чем при выполнении с опросом.

Достоинство системы прерывания заключается в том, что для обеспечения прерываний не нужно принимать никаких мер на уровне рабочей программы пользователя. Составляя свою программу, программист может даже ничего не знать о системе прерываний. Однако системному программисту возможно потребуется разработать особые программы обслуживания прерываний, специфичные для данного устройства, если таких программ может не оказаться в средствах ПО, созданных разработчиками этих средств.

6. Организация памяти ПК

6.1. Иерархии памяти

В основе реализации иерархии памяти современных компьютеров лежат два принципа: *принцип локальности обращений* и *соотношение стоимость/производительность*. Принцип локальности обращений говорит о том, что большинство программ к счастью не выполняют обращений ко всем

своим командам и данным равновероятно, а оказывают предпочтение некоторой части своего адресного пространства.

Иерархия памяти современных компьютеров строится на нескольких уровнях, причем более высокий уровень меньше по объему, быстрее и имеет большую стоимость в пересчете на байт, чем более низкий уровень. Уровни иерархии взаимосвязаны: все данные на одном уровне могут быть также найдены на более низком уровне, и все данные на этом более низком уровне могут быть найдены на следующем нижележащем уровне и так далее, пока мы не достигнем основания иерархии.

Иерархия памяти обычно состоит из многих уровней, но в каждый момент времени мы имеем дело только с двумя близлежащими уровнями. Минимальная единица информации, которая может либо присутствовать, либо отсутствовать в двухуровневой иерархии, называется блоком. Размер блока может быть либо фиксированным, либо переменным. Если этот размер зафиксирован, то объем памяти является кратным размеру блока.

Успешное или неуспешное обращение к более высокому уровню называются соответственно попаданием (hit) или промахом (miss). *Попадание* - есть обращение к объекту в памяти, который найден на более высоком уровне, в то время как промах означает, что он не найден на этом уровне. Доля попаданий (hit rate) или коэффициент попаданий (hit ratio) есть доля обращений, найденных на более высоком уровне. Иногда она представляется процентами. Доля промахов (miss rate) есть доля обращений, которые не найдены на более высоком уровне.

Поскольку повышение производительности является главной причиной появления иерархии памяти, частота попаданий и промахов является важной характеристикой. Время обращения при попадании (hit time) есть время обращения к более высокому уровню иерархии, которое включает в себя, в частности, и время, необходимое для определения того, является ли обращение попаданием или промахом. Потери на промах (miss penalty) есть время для замещения блока в более высоком уровне на блок из более низкого уровня плюс время для пересылки этого блока в требуемое устройство (обычно в процессор). Потери на промах далее включают в себя две компоненты: время доступа (access time) - время обращения к первому слову блока при промахе, и время пересылки (transfer time) - дополнительное время для пересылки оставшихся слов блока. Время доступа связано с задержкой памяти более низкого уровня, в то время как время пересылки связано с полосой пропускания канала между устройствами памяти двух смежных уровней.

Чтобы описать некоторый уровень иерархии памяти надо ответить на следующие четыре вопроса:

1. Где может размещаться блок на верхнем уровне иерархии? (размещение блока).

2. Как найти блок, когда он находится на верхнем уровне? (идентификация блока).

3. Какой блок должен быть замещен в случае промаха? (замещение блоков).

4. Что происходит во время записи? (стратегия записи).

6.2. Организация кэш-памяти

В разделе 4 было сказано несколько слов о кеш-памяти. Теперь более подробно познакомимся с ее организацией.

Рассмотрим организацию кэш-памяти более детально, отвечая на четыре вопроса об иерархии памяти.

Принципы размещения блоков в кэш-памяти определяют три основных типа их организации:

- Если каждый блок основной памяти имеет только одно фиксированное место, на котором он может появиться в кэш-памяти, то такая кэш-память называется кэшем с прямым отображением (direct mapped). Это наиболее простая организация кэш-памяти, при которой для отображение адресов блоков основной памяти на адреса кэш-памяти просто используются младшие разряды адреса блока. Таким образом, все блоки основной памяти, имеющие одинаковые младшие разряды в своем адресе, попадают в один блок кэш-памяти, т.е.

(адрес блока кэш-памяти) = (адрес блока основной памяти) mod (число блоков в кэш-памяти)

- Если некоторый блок основной памяти может располагаться на любом месте кэш-памяти, то кэш называется полностью ассоциативным (fully associative).
- Если некоторый блок основной памяти может располагаться на ограниченном множестве мест в кэш-памяти, то кэш называется множественно-ассоциативным (set associative). Обычно множество представляет собой группу из двух или большего числа блоков в кэше. Если множество состоит из n блоков, то такое размещение называется множественно-ассоциативным с n каналами (n-way set associative). Для размещения блока прежде всего необходимо определить множество. Множество определяется младшими разрядами адреса блока памяти (индексом):

(адрес множества кэш-памяти) = (адрес блока основной памяти) mod (число множеств в кэш-памяти)

Далее, блок может размещаться на любом месте данного множества.

Диапазон возможных организаций кэш-памяти очень широк: кэш-память с прямым отображением есть просто одноканальная множественно-

ассоциативная кэш-память, а полностью ассоциативная кэш-память с m блоками может быть названа m-канальной множественно-ассоциативной. В современных процессорах как правило используется либо кэш-память с прямым отображением, либо двух- (четырех-) канальная множественно-ассоциативная кэш-память.

У каждого блока в кэш-памяти имеется адресный тег, указывающий, какой блок в основной памяти данный блок кэш-памяти представляет. Эти теги обычно одновременно сравниваются с выработанным процессором адресом блока памяти.

Кроме того, необходим способ определения того, что блок кэш-памяти содержит достоверную или пригодную для использования информацию. Наиболее общим способом решения этой проблемы является добавление к тегу так называемого бита достоверности (valid bit).

Адресация множественно-ассоциативной кэш-памяти осуществляется путем деления адреса, поступающего из процессора, на три части: поле смещения используется для выбора байта внутри блока кэш-памяти, поле индекса определяет номер множества, а поле тега используется для сравнения. Если общий размер кэш-памяти зафиксировать, то увеличение степени ассоциативности приводит к увеличению количества блоков в множестве, при этом уменьшается размер индекса и увеличивается размер тега.

При возникновении промаха, контроллер кэш-памяти должен выбрать подлежащий замещению блок. Польза от использования организации с прямым отображением заключается в том, что аппаратные решения здесь наиболее простые. Выбирать просто нечего: на попадание проверяется только один блок и только этот блок может быть замещен. При полностью ассоциативной или множественно-ассоциативной организации кэш-памяти имеются несколько блоков, из которых надо выбрать кандидата в случае промаха. Как правило для замещения блоков применяются две основных стратегии: случайная и LRU.

В первом случае, чтобы иметь равномерное распределение, блоки-кандидаты выбираются случайно. В некоторых системах, чтобы получить воспроизводимое поведение, которое особенно полезно во время отладки аппаратуры, используют псевдослучайный алгоритм замещения.

Во втором случае, чтобы уменьшить вероятность выбрасывания информации, которая скоро может потребоваться, все обращения к блокам фиксируются. Заменяется тот блок, который не использовался дольше всех (LRU - Least-Recently Used).

Достоинство случайного способа заключается в том, что его проще реализовать в аппаратуре. Когда количество блоков для поддержания трассы увеличивается, алгоритм LRU становится все более дорогим и часто только приближенным.

При обращениях к кэш-памяти на реальных программах преобладают обращения по чтению. Все обращения за командами являются обращениями по чтению и большинство команд не пишут в память. Обычно операции записи

составляют менее 10% общего трафика памяти. Желание сделать общий случай более быстрым означает оптимизацию кэш-памяти для выполнения операций чтения, однако при реализации высокопроизводительной обработки данных нельзя пренебрегать и скоростью операций записи.

К счастью, общий случай является и более простым. Блок из кэш-памяти может быть прочитан в то же самое время, когда читается и сравнивается его тег. Таким образом, чтение блока начинается сразу как только становится доступным адрес блока. Если чтение происходит с попаданием, то блок немедленно направляется в процессор. Если же происходит промах, то от заранее считанного блока нет никакой пользы, правда нет и никакого вреда.

Однако при выполнении операции записи ситуация коренным образом меняется. Именно процессор определяет размер записи (обычно от 1 до 8 байтов) и только эта часть блока может быть изменена. В общем случае это подразумевает выполнение над блоком последовательности операций чтение-модификация-запись: чтение оригинала блока, модификацию его части и запись нового значения блока. Более того, модификация блока не может начинаться до тех пор, пока проверяется тег, чтобы убедиться в том, что обращение является попаданием. Поскольку проверка тегов не может выполняться параллельно с другой работой, то операции записи отнимают больше времени, чем операции чтения.

Очень часто организация кэш-памяти в разных машинах отличается именно стратегией выполнения записи. Когда выполняется запись в кэш-память имеются две базовые возможности:

- сквозная запись (write through, store through) - информация записывается в два места: в блок кэш-памяти и в блок более низкого уровня памяти.
- запись с обратным копированием (write back, copy back, store in) - информация записывается только в блок кэш-памяти. Модифицированный блок кэш-памяти записывается в основную память только когда он замещается. Для сокращения частоты копирования блоков при замещении обычно с каждым блоком кэш-памяти связывается так называемый бит модификации (dirty bit). Этот бит состояния показывает был ли модифицирован блок, находящийся в кэш-памяти. Если он не модифицировался, то обратное копирование отменяется, поскольку более низкий уровень содержит ту же самую информацию, что и кэш-память.

Оба подхода к организации записи имеют свои преимущества и недостатки. При записи с обратным копированием операции записи выполняются со скоростью кэш-памяти, и несколько записей в один и тот же блок требуют только одной записи в память более низкого уровня. Поскольку в этом случае обращения к основной памяти происходят реже, вообще говоря требуется меньшая полоса пропускания памяти, что очень привлекательно для мультипроцессорных систем. При сквозной записи промахи по чтению не

влияют на записи в более высокий уровень, и, кроме того, сквозная запись проще для реализации, чем запись с обратным копированием. Сквозная запись имеет также преимущество в том, что основная память имеет наиболее свежую копию данных. Это важно в мультипроцессорных системах, а также для организации ввода/вывода.

Когда процессор ожидает завершения записи при выполнении сквозной записи, то говорят, что он приостанавливается для записи (write stall). Общий прием минимизации остановов по записи связан с использованием буфера записи (write buffer), который позволяет процессору продолжить выполнение команд во время обновления содержимого памяти. Следует отметить, что остановы по записи могут возникать и при наличии буфера записи.

При промахе во время записи имеются две дополнительные возможности:

- разместить запись в кэш-памяти (write allocate) (называется также выборкой при записи (fetch on write)). Блок загружается в кэш-память, вслед за чем выполняются действия аналогичные выполняющимся при выполнении записи с попаданием. Это похоже на промах при чтении.
- не размещать запись в кэш-памяти (называется также записью в окружение (write around)). Блок модифицируется на более низком уровне и не загружается в кэш-память.

Обычно в кэш-памяти, реализующей запись с обратным копированием, используется размещение записи в кэш-памяти (в надежде, что последующая запись в этот блок будет перехвачена), а в кэш-памяти со сквозной записью размещение записи в кэш-памяти часто не используется (поскольку последующая запись в этот блок все равно пойдет в память).

Формула для среднего времени доступа к памяти в системах с кэш-памятью выглядит следующим образом:

```
Среднее время доступа = Время обращения при попадании + Доля промахов * Потери при промахе
```

Эта формула наглядно показывает пути оптимизации работы кэш-памяти: сокращение доли промахов, сокращение потерь при промахе, а также сокращение времени обращения к кэш-памяти при попадании.

6.3. Организация оперативной памяти (RAM)

6.3.1. Типы и классификация ОП

Оперативная память (ОП) — совокупность ОЗУ, объединенных в одну систему, управляемую процессором. Для обеспечения приспосабливаемости ЭВМ к конкретным потребностям пользователей применяют принцип блочного построения 0П. Так, например, на основе блоков 03У емкостью 128 и 256 Кслов можно построить ОП любой емкости. ОП заданной емкости, составленная из нескольких блоков ОЗУ, называется *многоблочной* 0П.

Функциональном отношении многоблочная ОП рассматривается как одно ОЗУ с емкостью, равной сумме емкостей блоков, и быстродействием, примерно равным быстродействию отдельного блока. Адрес ячеек такой 0П содержит адрес блока и адрес ячейки памяти в заданном блоке ОЗУ.

Устройства, подключенные к 0П, обращаются к ней независимо друг от друга. Принцип обслуживания запросов к ОП - приоритетный. Устройствам присваиваются приоритеты: низший — центральному процессору, более высший — ВЗУ. ОП обслуживает очередной запрос с наивысшим приоритетом, а все остальные запросы от других устройств ожидают момента окончания обслуживания. Такой принцип обслуживания объясняется тем, что ВЗУ не могут долго ждать, так как большое время ожидания приводит к потере информации, записываемой или считываемой с непрерывно движущегося носителя. ОП, ресурсы которой распределяются между несколькими потребителями, называют 0П с многоканальным доступом.

Многоблочная ОП, в которой допускается совместное выполнение нескольких обращений к разным блокам ОЗУ называется ОП с расслоением обращений. В такой ОП блоки ОЗУ функционируют параллельно во времени, что возможно, если последующие обращения к ОП адресованы к блокам, не запятым обслуживанием предшествующих запросов. Степень расслоения обращений характеризуется коэффициентом расслоения, равным среднему числу обращений к ОП, которые могут быть приняты на обслуживание одновременно. Чем выше коэффициент расслоения, тем выше производительность.

Из микросхем, памяти (RAM - Random Access Memory, память с произвольным доступом) используется два основных типа: *статическая* (SRAM - Static RAM) и *динамическая* (DRAM - Dynamic RAM).

Рассмотрим понятия статическая и динамическая. Назовем, упорядоченную последовательность информационных и управляющих слов образует массив. Количество ячеек памяти, используемое для представления массива в ЭВМ, называется длиной массива.

Группа ячеек памяти с последовательными номерами Аб, Аб+1, Аб+2, ... , Аб+n, представляющая массив длиной (n+1), рассматривается как массив ячеек памяти с базовым адресом Аб (рис. 6.1).

Статическое распределение памяти основано на выделении ячеек ОП для массивов в процессе анализа и составления программы, т. е. до начала решения задачи и при выполнении программы базисные адреса сохраняют постоянные значения.

При статическом распределении ячеек памяти в массивах память используется неэффективно, так как в процессе решения задачи количество слов в массиве в большинстве случаев меньше длины массива ячеек с базовым адресом Аб. Поэтому этот способ применяют лишь в простейших системах программирования на небольших ЭВМ.

Выделение в памяти массива ячеек длиной (n+1) и базовым адресом Аб

Адрес	Оперативная память
Аб Аб+1 Аб+2 … Аб+n	

Рис. 6.1.

Динамическое распределение памяти основано на выделении ячеек памяти для массивов с учетом их длин в порядке их появления в процессе решения задачи. Оно используется для экономии ячеек памяти в пределах одной программы и при мультипрограммной работе ЭВМ для распределения памяти между программами.

В статической памяти элементы (ячейки) построены на различных (вариантах триггеров - схем с двумя устойчивыми состояниями. После записи бита в такую ячейку она может пребывать в этом состоянии столь угодно долго - необходимо только наличие питания. При обращении к микросхеме статической памяти на нее подается полный адрес, который при помощи внутреннего дешифратора преобразуется в сигналы выборки конкретных ячеек. Ячейки статической памяти имеют малое время, срабатывания (единицы-десятки наносекунд), однако микросхемы на их основе имеют низкую удельную плотность данных (порядка единиц Мбит на корпус) и высокое энергопотребление. Поэтому статическая память используется в основном в качестве буферной (кэш-память).

В динамической памяти ячейки построены на основе областей с накоплением зарядов, занимающих гораздо меньшую площадь, нежели 1Н1)иггеры, ч практически не потребляющих энергии при хранении. При записи бита в такую ячейку в ней (формируется электрический заряд, который сохраняется в течение нескольких миллисекунд; для постоянного сохранения заряда ячейки необходимо регенерировать - перезаписывать содержимое для восстановления 'зарядов. Ячейки микросхем динамической памяти организованы в виде прямоугольной (обычно - квадратной) матрицы; при обращении к микросхеме на ее входы вначале подается адрес строки матрицы, сопровождаемый сигналом RAS (Row Address Strobe - строб адреса строки), затем, через некоторое время - адрес столбца, сопровождаемый сигналом CAS (Column Address Strobe - строб адреса столбца). При каждом обращении к ячейке регенерируют все ячейки выбранной строки, поэтому для полной регенерации матрицы достаточно перебрать адреса строк. Ячейки

динамической памяти имеют большее время срабатывания (десятки- сотни наносекунд), но большую удельную плотность (порядка десятков Мбит на корпус) и меньшее энергопотребление. Динамическая память используется в качестве основной.

Обычные виды SRAM и DRAM называют также *асинхронными,* потому что установка адреса, подача управляющих сигналов и чтение/запись данных могут выполняться в произвольные моменты времени - необходимо только соблюдение временных соотношений между этими сигналами. В эти временные соотношения включены так называемые охранные интервалы, необходимые для стабилизации сигналов, которые не позволяют достичь теоретически возможного быстродействия памяти. Существуют также синхронные виды памяти, получающие внешний синхросигнал, к импульсам которого жестко привязаны моменты подачи адресов и обмена данными; помимо экономии времени на охранных интервалах, они позволяют более полно использовать внутреннюю конвейеризацию и блочный доступ.

FPM DRAM (Fast Page Mode DRAM - динамическая память с быстрым страничным доступом) активно используется в последние несколько лет. Память со страничным доступом отличается от обычной динамической памяти тем, что после выбора строки матрицы и удержании RAS допускает многократную установку адреса столбца, стробируемого CAS, и также быструю регенерацию по схем.е "CAS прежде RAS". Первое позволяет ускорить блочные передачи, когда весь блок данных или его часть находятся внутри одной строки матрицы, называемой в этой системе страницей, а второе - снизить накладные расходы на регенерагщю памяти.

ED0 (Entended Data Out - расширенное время удержания данных на выходе) фактически представляют собой обычные микросхемы FPM, на выходе которых установлены регистры-защелки данных. При страничном обмене такие микросхемы работают в режиме простого конвейера: удерживают на выходах данных содержимое последней выбранной ячейки, в то время как на их входы уже подается адрес следующей выбираемой ячейки. Это позволяет примерно на 15% по сравнению с FPM ускорить процесс считывания последовательных массивов данных. При случайной адресации такая память ничем не отличается от обычной.

BEDO (Burst EDO - EDO с блочным доступом) - память на основе EDO, работающая не одиночными, а пакетными циклами чтения/записи. Современные процессоры, благодаря внутреннему и внешнему кэшированию команд и данных, обмениваются с основной памятью преимущественно блоками слов максимальной ширины. В случае памяти BEDO отпадает необходимость постоянной подачи последовательных адресов на входы микросхем с соблюдением необходимых временных задержек - достаточно стробировать переход к очередному слову отдельным сигналом.

SDRAM (Synchronous DRAM - синхронная динамическая память) -память с синхронным доступом, работающая быстрее обычной асинхронной

(EPM/EDO/BEDO). Помимо синхронного метода доступа, SDRAM использует внутреннее разделение массива памяти на два независимых банка, что позволяет совмещать выборку из одного банка с установкой адреса в другом банке. SDRAM также поддерживает блочный обмен. Основная выгода от использования SDRAM состоит в поддержке последовательного доступа в синхронном режиме, где не требуется дополнительных тактов ожидания. При случайном доступе SDRAM работает практически с той же скоростью, что и FPM/EDO.

PB SRAM (Pipelined Burst SRAM - статическая память с блочным конвейерным, доступом) - разновидность синхронных SRAM с внутренней конвейеризацией, за счет которой примерно вдвое повышается скорость обмена опоками данных.

Микросхемы памяти имеют четыре основные характеристики - тип, объем, структуру и время доступа. Тип обозначает статическую или динамическую память, объем показывает общую емкость микросхемы, а структура - количество ячеек памяти и разрядность каждой ячейки. Например, 28/32- выводные DIP- микросхемы SRAM имеют восьмиразрядную структуру (8k*8, 16k*8, 32k*8, 64k*8, 128k*8), и кэш для 486 объемом 256 кб будет состоять из восьми микросхем 32k*S или четырех микросхем 64k*8 (речь идет об области данных дополнительные микросхемы для хранения признаков (tag) могут иметь другую структуру). Две микросхемы по 128k*8 поставить уже нельзя, так как нужна 32- разрядная шина данных, что могут дать только четыре параллельных микросхемы. Распространенные PB SRAM в 100-выводных корпусах PQFP имеют 32-разрядную структуру 32k*32 или 64k*32 и используются по две или по четыре в платах для Pentuim.

Аналогично, 30-контактные SIMM имеют 8-разрядную структуру и ставятся с процессорами 286, 386SX и 486SLC по два, а с 386DX, 486DLC и обычными 486 - по четыре. 72-контактные SIMM имеют 32-разрядную структуру и могут ставиться с 486 по одному, а с Pentium и Pentium Pro -по два. 168- контактные DIMM имеют 64- разрядную структуры и ставятся в Pentium и Pentium Pro no одному. Установка модулей памяти или микросхем кэша в количестве больше минимального позволяет некоторым платам ускорить работу с ними, используя принцип /расслоения (Interleave - чередование).

Время доступа характеризует скорость работы микросхемы и обычно указывается в наносекундах через тире в конце наименования. На более медленных динамических микросхемах могут указываться только первые цифры (-7 вместо -70, -15 вместо -150), на более быстрых статических "-15" или "-20" обозначают реальное время доступа к ячейке. Часто на микросхемах указывается минимальное из всех возможных времен доступа - например, распространена маркировка 70 нс EDO DRAM, как 50, или 60 нс - как 45, хотя такой цикл достижим только в блочном. режиме, а в одиночном режиме микросхема по-прежнему срабатывает за 70 или 60 не. Аналогичная ситуация имеет место в маркировке PB SRAM: 6 нс вместо 12, и 7 - вместо 15.

Микросхемы SDRAM обычно маркируются временем доступа в блочном режиме (10 или 12 нс).

Тип модуля памяти DIP (Dual In line Package - корпус с двумя рядами выводов) - классические микросхемы, применявшиеся в блоках основной памяти XT и ранних AT, а сейчас - в блоках кэш-памяти. SIP (Single In line Package - корпус с одним рядом выводов) - микросхема с одним рядом выводов, устанавливаемая вертикально. SIPP (Single In line Pinned Package - модуль с одним рядом проволочных выводов) - модуль памяти, вставляемый в панель наподобие микросхем DIP/SIP; применялся в ранних A T.

SIMM (Single In line Memory Module - модуль памяти с одним рядом контактов) - модуль памяти, вставляемый в зажимающий разъем; применяется во всех современных платах, а также во многих адаптерах, принтерах и прочих устройствах. SIMM имеет контакты с двух сторон модуля, но все они соединены между собой, образуя как бы один ряд контактов.

DIMM (Dual In line Memory Module - модуль памяти с двумя рядами контактов) - модуль памяти, похожий на SIMM, но с раздельными контактами (обычно 2 x 84), за счет чего увеличивается разрядность или число банков памяти в модуле. Применяется в основном в компьютерах Apple ч новых платах P5 и P6.

На SIMM в настоящее время устанавливаются преимущественно микросхемы FPM/EDO/BEDO, а на DIMM - EDO/BEDO/SDRAM.

CELP (Card Egde Low Profile - невысокая карта с ножевым разъемом на краю) - модуль внешней кэш-памяти, собранный на микросхемах SRAM (асинхронный) или PB SRAM (синхронный). По внешнему виду похож на 72-контактный SIMM, имеет емкость 256 или 512 кб. Другое название -COAST (Cache On A Stick - буквально "кэш на палочке").

Модули динамической памяти, помимо памяти для данных, могут. иметь дополнительную память для хранения битов четности (Parity) для байтов данных - такие SIMM иногда называют 9- и 36- разрядными модулями (по одному биту четности на байт данных). Биты четности служат ()л.я контроля правильности считывания данных из модуля, позволяя обнаружить часть ошибок (но не все ошибки). Модули с четностью имеет смысл применять лишь там, где нуж,на очень высокая надежность - для обычных применений подходят и тщательно проверенные модули без четности, при условии, что системная плата поддерживает такие типы модулей.

Проще всего определить тип модуля по маркировке и количеству микросхем памяти на нем: например, если на 30-контактном SIMM две микросхемы одного типа и одна - другого, то две первых содержат (данные (каждая - по четыре разряда), а третья - биты четности (она одноразрядная). В 72- контактном SIMM с двенадцатью микросхемами восемь из них хранят данные, а четыре - биты четности. Модули с количеством микросхем 2, 4 или 8 не имеют памяти под четность.

Иногда на модули ставится так называемый имитатор четности - микросхема- сумматор, выдающая при считывании ячейки всегда правильный бит четности. В основном это предназначено для установки таких модулей в платы, где проверка четности не отключается; однако, существуют модули, где такой сумматор маркирован как "честная" микросхема памяти.

72-контактные SIMM имеют четыре специальных линии PD (Presence Deled - обнаружение наличия), на которых при помощи перемычек может быть установлено до 16 комбинаций сигналов. Линии PD используются некоторыми "Brand name"- платами для определения наличия модулей в разъемах и их параметров (объеми и быстродействия). Большинство универсальных плат производства "третьих фирм", как их выпускаемые ими SIMM, не используют линий PD.

В модулях DIMM, в соответствии со спецификацией JEDEC, технология PD реализуется при помощи перезаписываемого ПЗУ с последовательным доступом (Serial EEPROM) и носит название Serial Presence Detect (SPD). ПЗУ представляет собой 8- выводную микросхему, размещенную в углу платы DIMM, а его содержимое описывает конфигурацию и параметры модуля. Системные платы с чипсетами 440LX/BX могут использовать SPD для настройки системы управления памятью. Некоторые системные платы могут обходиться без SPD, определяя конфигурацию модулей обычным путем - это стимулирует выпуск рядом производителей DIMM без ПЗУ, не удовлетворяющих спецификации JEDEC.

6.3.2. Адресация информации и обработка адресов

Для расширения операционных возможностей и увеличения производительности процессора применяются различные способы адресации информации, отличающиеся порядком использования и обработки адресного поля в команде, посредством которого организуется доступ к информации, хранящейся в оперативной памяти (или в ПЗУ) ЭВМ.

Рассмотрим способы адресации операндов и команд.

6.3.2.1. Непосредственная адресация

В коде команды (в коде одного или нескольких адресов) размещается непосредственный операнд, если число значащих цифр операнда не превышает длины адресной части команды. Такая адресация используется для хранения различного рода констант и находит широкое применение в универсальных ЭВМ в целях экономии ячеек ОП и уменьшения времени выполнения команды.

6.3.2.2. Прямая адресация

Исполнительный адрес - адрес ячейки ОП, в которой хранится адресуемое слово, совпадает с адресной частью команды. Этот метод используется в ЭВМ в комбинации с другими методами адресации.

6.3.2.3. Прямая регистровая адресация

В адресном поле команды содержится адрес R регистра СОЗУ процессора, в котором хранится операнд (рис. 6.2.).

Прямая регистровая адресация

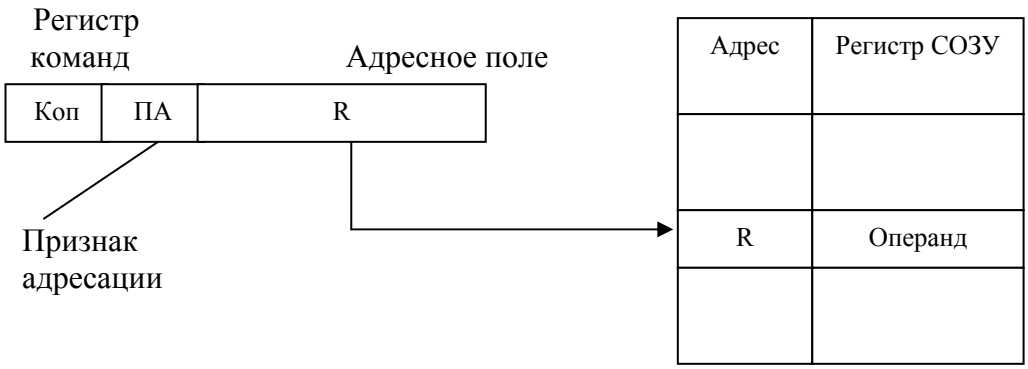


Рис. 6.2.

При таком способе адресации быстродействие ЭВМ повышается, так как нет необходимости извлекать операнды из ОП и команда имеет более короткий формат, так как регистров в СОЗУ обычно значительно меньше, чем ячеек в оперативной памяти. Прямая регистровая адресация используется для операндов, многократно используемых при выполнении программ.

6.3.2.4. Подразумеваемая адресация

При такой адресации в команде не содержится явных указаний об адресе операнда участвующего в операции, или адреса, по которому передается результат операции. Этот адрес подразумевается и фактически задается кодом операции. Например, в одноадресных командах адресом второго операнда или результата операции подразумевается адрес — содержимое специального регистра процессора, хранящего второй операнд или принимающего результат операции; в двухадресных командах подразумевается помещение операции по адресу одного из операндов.

6.3.2.5. Косвенная адресация

В адресном поле команды указывается адрес ячейки оперативной памяти, содержащей другой адрес, который может быть исполнительным Аисп или еще одним косвенным адресом (так называемая многоступенчатая косвенная адресация). Таким образом, косвенная адресация (рис. 6.3) может быть определена как «адресация адреса».

Косвенная адресация с использованием оперативной памяти

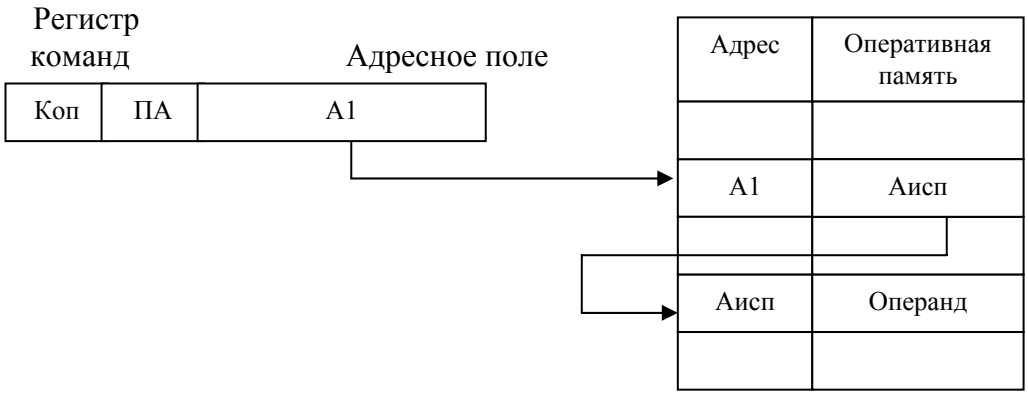


Рис. 6.3.

Она используется в тех случаях, когда число разрядов в адресной части команды недостаточно для указания всех адресов оперативной памяти ЭВМ.

6.3.2.6. Косвенная регистровая адресация

При таком способе адресации в адресном поле команды число является адре-сом R регистра Рсозу, который содержит исполнительный адрес Аисп (рис. 6.4.).

Косвенная регистровая адресация

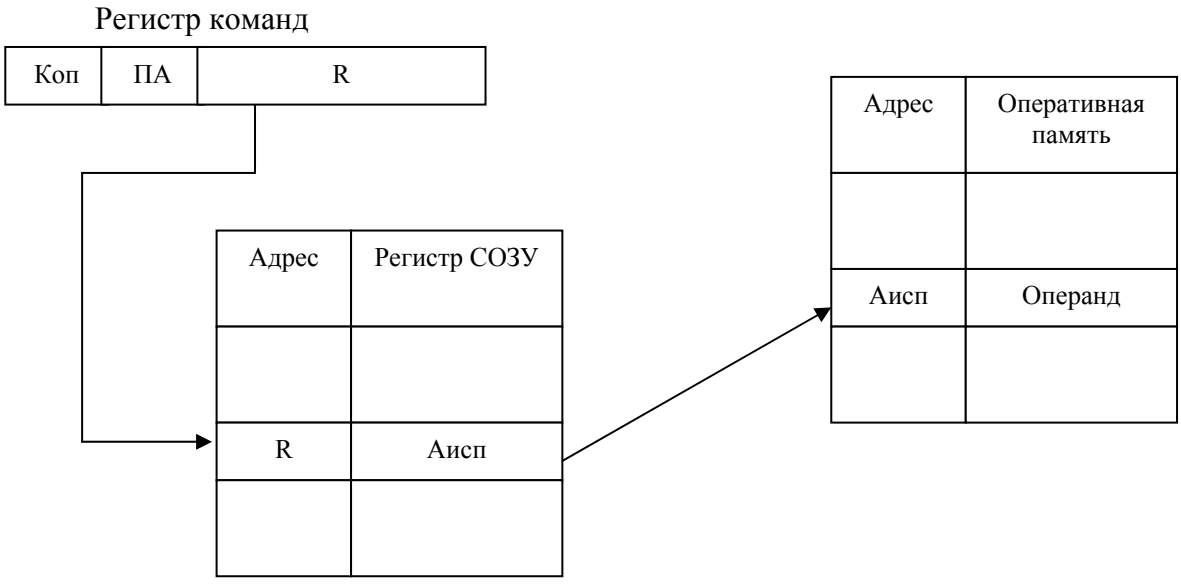


Рис. 6.4.

При такой адресации необходимо сначала загрузить регистр R, а поэтому ее используют тогда, когда программа многократно использует один и тот же адрес ячейки.

6.3.2.7. Модификация адресов

Рассмотренные методы адресации обеспечивают адресацию переменных и констант. При решении ряда задач на ЭВМ необходимо выполнять некоторые участки программ многократно (цикличность вычислительного процесса) над различными операндами, расположенными упорядоченно в массивах ОП. Поскольку операнды, обрабатываемые при повторениях цикла, имеют разные адреса, то каждый цикл в программе можно представить виде последовательности команд, отличающихся адресными частями. Однако при таком подходе программа решения задачи оказывается слишком длинной и ее составление чрезмерно трудоемким.

Программирование вычислительных циклов существенно упрощается, если после каждого цикла обеспечить автоматическое изменение в соответствующих командах их адресных частей. Процедура изменения адреса в командах называется модификацией адреса. Модификация адресов команд основана на возможности выполнения над кодами команд или их частями арифметических и логических операций. В качестве операндов в командах вычислительного цикла могут фигурировать элементы массивов называемые переменными с индексами. Элемент массива представляется базовым адресом Аб и индекса i, указывающего, на сколько единиц должен быть изменен адрес команды перед ее выполнением.

Программный способ модификации адресов в команде значительно замедляет процесс обработки переменных с индексом и требует для этих целен большой емкости оперативной памяти. В связи с этим в современных ЭВМ для модификации адресов используют аппаратные средства. В этом случае адрес в команде (рис. 6.5) представляется двумя полями.

В поле В указывается базовый адрес массива Аб оперативной памяти. Поле Х называется индексом. Если Х=0, то адрес Аб не модифицируется, т.е. является исполнительным Аисп. Значение Х<>0 определяет адрес ячейки памяти индексов, в которой хранится индекс i. Модификация адреса сводится к вычислению исполнительного адреса Аисп=Аб+(Х), где (Х) — содержимое ячейки Х индексной памяти.

В качестве индексной памяти используют в процессоре так называемые индексные регистры СОЗУ. Суммирование производится или АЛБ процессора, пли в специальном сумматоре обработки адресов, что несколько только увеличивает объем процессора.

Индексная адресация с использованием регистров

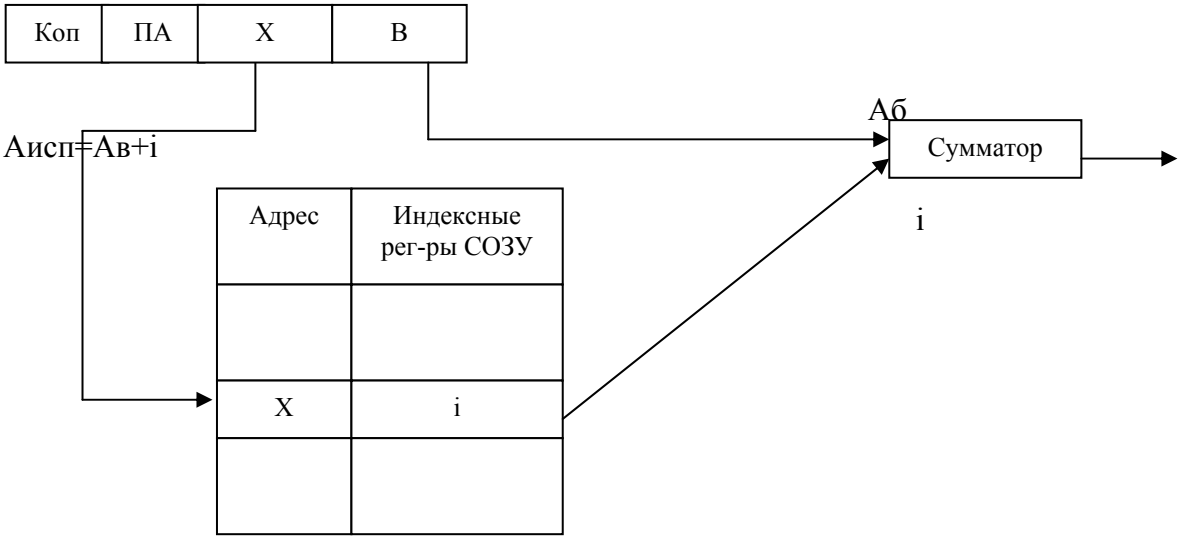


Рис. 6.5.

6.3.2.8. Относительная адресация

При динамическом распределении памяти базовые адреса массивов изменяются в процессе выполнения программы, в результате адреса не могут быть зафиксированы в программе. Для обеспечения динамического распределения памяти используют способ относительной адресации. Относительный адрес (рис. 6.6, а) состоит из двух полей: B, указывающего базовый адрес Аб массива D, представляющего собой относительный адрес Аотн. Поле D принято называть смещением D операнда относительно начала массива.

Исполнительный адрес вычисляется по формуле Аисп= (B)+D. Для хранения базовых адресов в целях повышения быстродействия ЭВМ используют также так называемые базовые регистры СОЗУ.

При формировании Аисп (рис. 6.6, а) на суммирование в SM затрачивается некоторое время. В целях уменьшения этого времени используют так называемый метод совмещения. В этом случае в базовом регистре B (рис. 6.6, б) содержатся старшие разряды, а в поле C) записывают младшие разряды исполнительного адреса Аисп, которые выдаются непосредственно в регистр адреса оперативной памяти (РгАОП). При совмещении, очевидно, базовый адрес Аб не может принимать значение адреса любой ячейки ОП, а только тех адресов, которые содержат в младших разрядах нули, соответствующие количеству разрядов поля D.

Относительная адресация

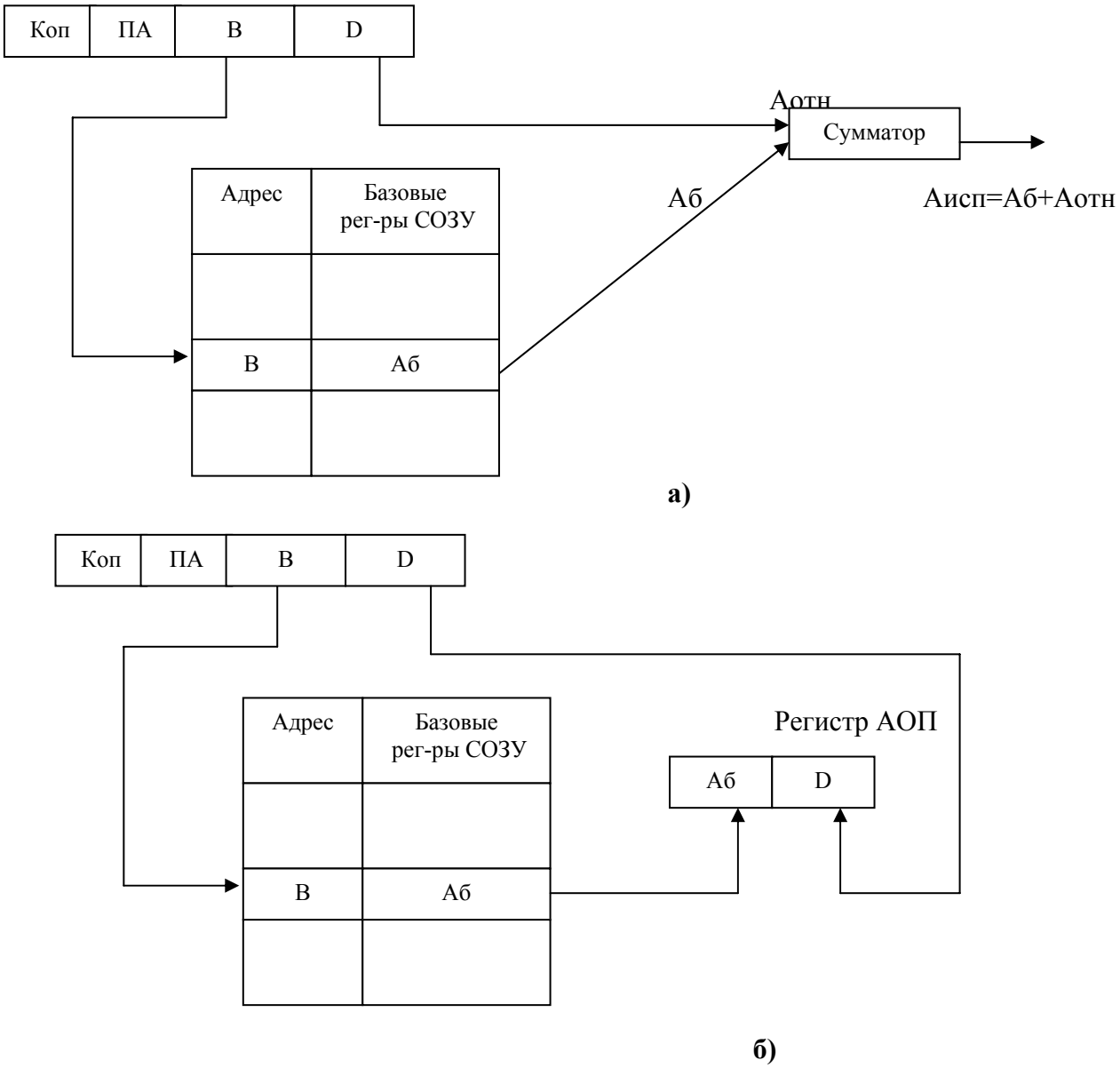


Рис. 6.6.

В универсальных ЭВМ используют совместно относительную адресацию и модификацию адресов (рис. 6.7). В этом случае Аисп вычисляется по формуле Аисп=(B)+(X)+D, где (B) — базовый адрес Аб (содержимое ячейки B), (X) — индекс i (содержимое ячейки X); D — смещение операнда (относительный адрес).

Формирование исполнительного адреса при относительной и индексной адресации

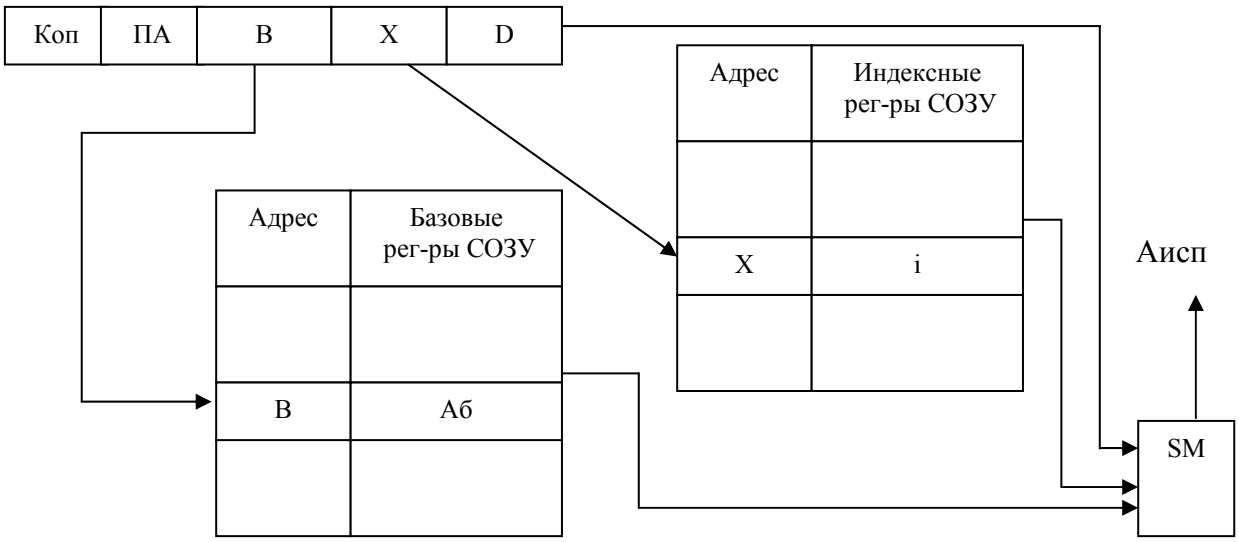


Рис. 6.7.

6.4. Организация виртуальной памяти

Обычно при решении задач на ЭВМ только часть информации размещается во внутренней памяти, а остальная хранится во внешней памяти. Таким образом, программист имеет дело с многоуровневой памятью и, планируя процесс решения задачи, включает в программу операции, вызывающие обмен информацией между различными ЗУ. Даже при наличии систем автоматизации программирования и хорошо организованных систем управления данными программирование задач для ЭВМ с многоуровневой памятью отличается сложностью и требует от исполнителей высокой квалификации. Особенно сложны процедуры обмена информацией между уровнями памяти, необходимые для организации мультипрограммной работы ЭВМ в режиме разделения времени. В связи с этим. в современных ЭВМ осуществляется автоматическое (не предусмотренное программой) планирование передач информации в многоуровневой памяти, основанное на построении виртуальной (фиктивной, кажущейся) одноуровневой памяти.

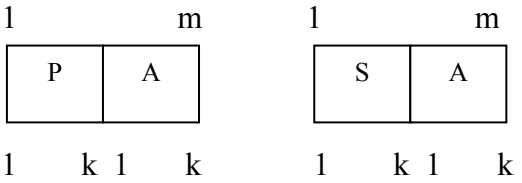
Под *виртуальной памятью* понимается совокупность ячеек всех ОЗУ и ВЗУ, имеющих сквозную нумерацию 0, 1, 2, ..., (Q-1). Программирование процессов решения задач производится в терминах математических (виртуальных) адресов 0.1,2,..., (Q-1). При этом команды ссылаются на математические адреса в предположении, что слово, идентифицированное любым математическим адресом, является доступным для процессора. Таким образом, для программиста создается одноуровневая память емкостью Q слов. В физическом отношении виртуальная память — это совокупность оперативных и внешних ЗУ, охваченных средствами преобразования

математических адресов в физические (действительные) адреса ячеек и автоматизации перемещения информации между устройствами памяти.

6.4.1. Страничная адресация памяти

Процессы преобразования адресов и перемещения информации наиболее просто реализуются при страничной адресации памяти. Метод страничной адресации состоит в том, что виртуальная память (множество адресов) разделяется на страницы емкостью 2^K соседних адресов. Так, к странице с адресом 0 относятся адреса 0, 1, 2, ... , (2^K-1) к странице с адресом $(1-2^K),(2^K+1),...(2^{K+1}-1)$ и так далее. В результате адрес слова будет состоять из двух полей P, указывающих адрес страницы, и A — адрес слова в странице P (рис. 6.8, а).

Адреса при страничной адресации



а) Виртуальный адрес б) Физический адрес

Рис. 6.8.

Если физическую память разделить, а сегменты, состоящие из 2^K соседних ячеек, то физические адреса в пределах одной виртуальной страницы по структуре будут полностью совпадать с математическими адресами (рис. 6.8, б), где S - адрес сегмента, а A - адрес слова (ячейки) в сегменте S. Размер страниц составляет 512-1024 слова, но в некоторых случаях возникает необходимость в использовании страниц размером 32—128 слов.

В процессе решения задачи страницы перемещаются между ОЗУ и ВЗУ. Если вычислительный процесс распределяется на страницу P, то она вызывается в ОЗУ. Когда надобность в информации, размещенной на странице P, отпадает, то она удаляется из ОЗУ в виртуальную память, освобождая место для других страниц. В результате перемещения граница P может быть помещена на любом сегменте S ОП.

Текущее состояние памяти ЭВМ характеризуется таблицей страниц (рис. 6.9). Отдельной странице виртуальной памяти Pi (i=1, 2, ..., Q-1) соответствует одна строка таблицы, в которой указываются параметры страницы Pi: Si — адрес сегмента ОЗУ, в котором размещается страница Pi, иначе говоря, физический адрес страницы Pi; di - признак доступности страницы: при di=1 страница Pi, хранится в ОЗУ и недоступна для центрального процессора.

Порядок использования таблицы страниц

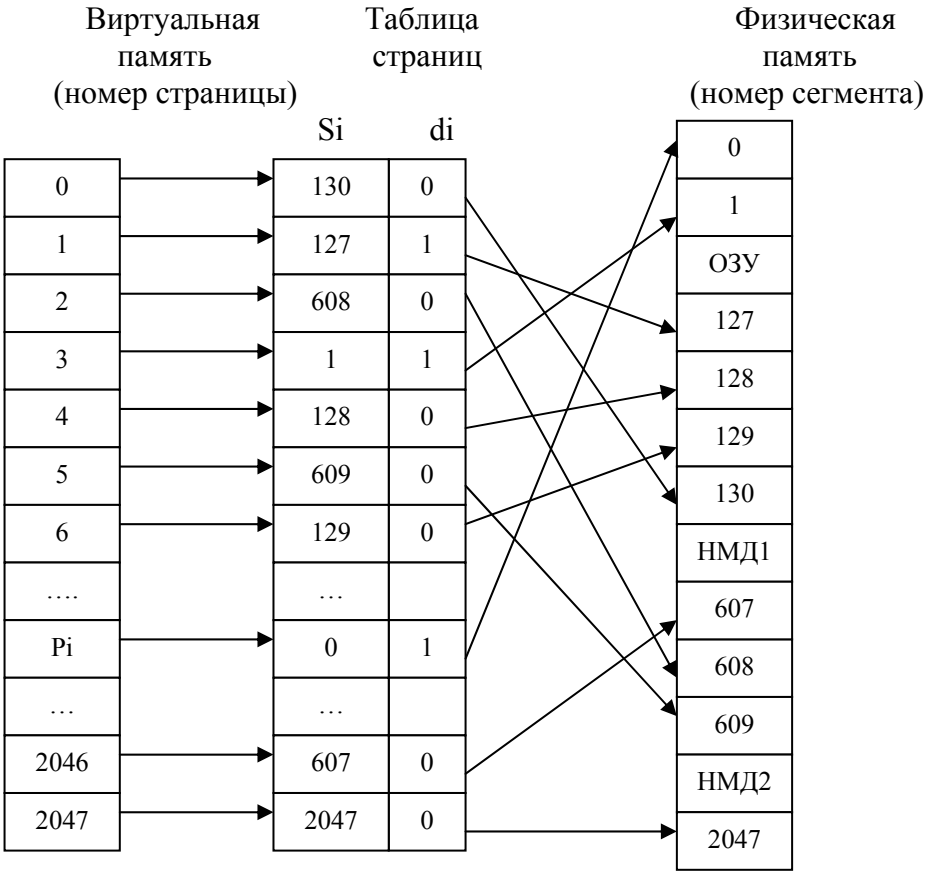


Рис. 6.9.

В таблице страниц также указываются параметры, определяющие страницу, которую надо удалить из ОЗУ (на рис 6.9 эти параметры не показаны) вследствие относительной низкой вероятности ее использования. Таблица страниц размещается в ОЗУ и в любое время доступна ЦП. Как видно из рис. 6.9, 2048 страниц виртуальной памяти могут размещается в 128 сегментах ОЗУ и на магнитных дисках НМД1 и НМД2 емкостью 960 страниц каждый.

Каждой странице Pi (i=0, 1, 2, . . ., 2047) виртуальной памяти соответствует свой сегмент, адрес которого определен в таблице страниц, и, следовательно, каждому слову присвоен свой физический адрес.

Операционная система (ОС) обеспечивает преобразование математических адресов в физические адреса.

6.4.2. Сегментация памяти

Другой подход к организации памяти опирается на тот факт, что программы обычно разделяются на отдельные области-сегменты. Каждый сегмент представляет собой отдельную логическую единицу информации, содержащую совокупность данных или программ и расположенную в адресном

пространстве пользователя. Сегменты создаются пользователями, которые могут обращаться к ним по символическому имени. В каждом сегменте устанавливается своя собственная нумерация слов, начиная с нуля.

Обычно в подобных системах обмен информацией между пользователями строится на базе сегментов. Поэтому сегменты являются отдельными логическими единицами информации, которые необходимо защищать, и именно на этом уровне вводятся различные режимы доступа к сегментам. Можно выделить два основных типа сегментов: программные сегменты и сегменты данных (сегменты стека являются частным случаем сегментов данных). Поскольку общие программы должны обладать свойством повторной входимости, то из программных сегментов допускается только выборка команд и чтение констант. Запись в программные сегменты может рассматриваться как незаконная и запрещаться системой. Выборка команд из сегментов данных также может считаться незаконной и любой сегмент данных может быть защищен от обращений по записи или по чтению.

Для реализации сегментации было предложено несколько схем, которые отличаются деталями реализации, но основаны на одних и тех же принципах.

В системах с сегментацией памяти каждое слово в адресном пространстве пользователя определяется виртуальным адресом, состоящим из двух частей: старшие разряды адреса рассматриваются как номер сегмента, а младшие - как номер слова внутри сегмента. Наряду с сегментацией может также использоваться страничная организация памяти. В этом случае виртуальный адрес слова состоит из трех частей: старшие разряды адреса определяют номер сегмента, средние - номер страницы внутри сегмента, а младшие - номер слова внутри страницы.

Как и в случае страничной организации, необходимо обеспечить преобразование виртуального адреса в реальный физический адрес основной памяти. С этой целью для каждого пользователя операционная система должна сформировать таблицу сегментов. Каждый элемент таблицы сегментов содержит описатель (дескриптор) сегмента (поля базы, границы и индикаторов режима доступа). При отсутствии страничной организации поле базы определяет адрес начала сегмента в основной памяти, а граница - длину сегмента. При наличии страничной организации поле базы определяет адрес начала таблицы страниц данного сегмента, а граница - число страниц в сегменте. Поле индикаторов режима доступа представляет собой некоторую комбинацию признаков блокировки чтения, записи и выполнения.

Таблицы сегментов различных пользователей операционная система хранит в основной памяти. Для определения расположения таблицы сегментов выполняющейся программы используется специальный регистр защиты, который загружается операционной системой перед началом ее выполнения. Этот регистр содержит дескриптор таблицы сегментов (базу и границу), причем база содержит адрес начала таблицы сегментов выполняющейся программы, а граница - длину этой таблицы сегментов. Разряды номера сегмента

виртуального адреса используются в качестве индекса для поиска в таблице сегментов. Таким образом, наличие базово-граничных пар в дескрипторе таблицы сегментов и элементах таблицы сегментов предотвращает возможность обращения программы пользователя к таблицам сегментов и страниц, с которыми она не связана. Наличие в элементах таблицы сегментов индикаторов режима доступа позволяет осуществить необходимый режим доступа к сегменту со стороны данной программы. Для повышения эффективности схемы используется ассоциативная кэш-память.

Отметим, что в описанной схеме сегментации таблица сегментов с индикаторами доступа предоставляет всем программам, являющимся частями некоторой задачи, одинаковые возможности доступа, т. е. она определяет единственную область (домен) защиты. Однако для создания защищенных подсистем в рамках одной задачи для того, чтобы изменять возможности доступа, когда точка выполнения переходит через различные программы, управляющие ее решением, необходимо связать с каждой задачей множество доменов защиты. Реализация защищенных подсистем требует разработки некоторых специальных аппаратных средств. Рассмотрение таких систем, которые включают в себя кольцевые схемы защиты, а также различного рода мандатные схемы защиты, выходит за рамки данного пособия.

7. Организация ввода-вывода

В вычислительной системе, состоящей из множества подсистем, необходим механизм для их взаимодействия. Эти подсистемы должны быстро и эффективно обмениваться данными. Например, процессор, с одной стороны, должен быть связан с памятью, с другой стороны, необходима связь процессора с устройствами ввода/вывода.

В современных ПК такой механизм можно разделить на несколько уровней:

- BIOS;
- Системные и локальные шины;
- Шины ввода/вывода.

7.1. BIOS

BIOS (Basic Input/Output System) - основная система ввода/вывода, зашитая в ПЗУ (отсюда название ROM BIOS). Она представляет собой набор программ проверки и обслуживания аппаратуры компьютера, и выполняет роль посредника между DOS и аппаратурой. BIOS получает управление при включении и сбросе системной платы, тестирует саму плату и основные блоки компьютера - видеоадаптер, клавиатуру, контроллеры дисков и портов ввода/вывода, настраивает Chipset платы и загружает внешнюю операционную систему. При работе под DOS, Windows BIOS управляет основными

устройствами, при работе под OS/2, UNIX, WinNT BIOS практически не используется, выполняя лишь начальную проверку и настройку.

Обычно на системной плате установлено только ПЗУ с системным (Main, System) BIOS, отвечающим за саму плату и контроллеры FDD, HDD, портов и клавиатуры; в системный BIOS практически всегда входит System Setup - программа настройки системы. Видеоадаптеры и контроллеры HDD с интерфейсом ST- 506 (MFM) и SCSI имеют собственные BIOS в отдельных ПЗУ; их также могут иметь и другие платы - интеллектуальные контроллеры дисков и портов, сетевые карты и т.п.

7.2. Системные и локальные шины

Одним из простейших механизмов, позволяющих организовать взаимодействие различных подсистем, является единственная *центральная шина*, к которой подсоединяются все подсистемы. Доступ к такой шине разделяется между всеми подсистемами. Подобная организация имеет два основных преимущества: низкая стоимость и универсальность. Поскольку такая шина является единственным местом подсоединения для разных устройств, новые устройства могут быть легко добавлены, и одни и те же периферийные устройства можно даже применять в разных вычислительных системах, использующих однотипную шину. Стоимость такой организации получается достаточно низкой, поскольку для реализации множества путей передачи информации используется единственный набор линий шины, разделяемый множеством устройств.

Главным недостатком организации с единственной шиной является то, что шина создает узкое горло, ограничивая, возможно, максимальную пропускную способность ввода/вывода. Если весь поток ввода/вывода должен проходить через центральную шину, такое ограничение пропускной способности весьма реально. В коммерческих системах, где ввод/вывод осуществляется очень часто, а также в суперкомпьютерах, где необходимые скорости ввода/вывода очень высоки из-за высокой производительности процессора, одним из главных вопросов разработки является создание системы нескольких шин, способной удовлетворить все запросы.

Одна из причин больших трудностей, возникающих при разработке шин, заключается в том, что максимальная скорость шины главным образом лимитируется физическими факторами: длиной шины и количеством подсоединяемых устройств (и, следовательно, нагрузкой на шину). Эти физические ограничения не позволяют произвольно ускорять шины. Требования быстродействия (малой задержки) системы ввода/вывода и высокой пропускной способности являются противоречивыми. В современных крупных системах используется целый комплекс взаимосвязанных шин, каждая из которых обеспечивает упрощение взаимодействия различных подсистем, высокую пропускную способность, избыточность (для увеличения отказоустойчивости) и эффективность.

Традиционно шины делятся на шины, обеспечивающие организацию связи процессора с памятью, и шины ввода/вывода. Шины ввода/вывода могут иметь большую протяженность, поддерживать подсоединение многих типов устройств, и обычно следуют одному из шинных стандартов. Шины процессор-память, с другой стороны, сравнительно короткие, обычно высокоскоростные и соответствуют организации системы памяти для обеспечения максимальной пропускной способности канала память-процессор. На этапе разработки системы, для шины процессор-память заранее известны все типы и параметры устройств, которые должны соединяться между собой, в то время как разработчик шины ввода/вывода должен иметь дело с устройствами, различающимися по задержке и пропускной способности.

Как уже было отмечено, с целью снижения стоимости некоторые компьютеры имеют единственную шину для памяти и устройств ввода/вывода. Такая шина часто называется *системной*. Персональные компьютеры, как правило, строятся на основе одной системной шины в стандартах ISA, EISA или MCA. Необходимость сохранения баланса производительности по мере роста быстродействия микропроцессоров привела к двухуровневой организации шин в персональных компьютерах на основе локальной шины. Локальной шиной называется шина, электрически выходящая непосредственно на контакты микропроцессора. Она обычно объединяет процессор, память, схемы буферизации для системной шины и ее контроллер, а также некоторые вспомогательные схемы. Типичными примерами локальных шин являются VL-Bus и PCI.

Рассмотрим типичную *транзакцию* на шине. Шинная транзакция включает в себя две части: посылку адреса и прием (или посылку) данных. Шинные транзакции обычно определяются характером взаимодействия с памятью: транзакция типа "Чтение" передает данные из памяти (либо в ЦП, либо в устройство ввода/вывода), транзакция типа "Запись" записывает данные в память. В транзакции типа "Чтение" по шине сначала посылается в память адрес вместе с соответствующими сигналами управления, индицирующими чтение. Память отвечает, возвращая на шину данные с соответствующими сигналами управления. Транзакция типа "Запись" требует, чтобы ЦП или устройство в/в послало в память адрес и данные и не ожидает возврата данных. Обычно ЦП вынужден простаивать во время интервала между посылкой адреса и получением данных при выполнении чтения, но часто он не ожидает завершения операции при записи данных в память.

Разработка шины связана с реализацией ряда дополнительных возможностей. Решение о выборе той или иной возможности зависит от целевых параметров стоимости и производительности. Первые три возможности являются очевидными: раздельные линии адреса и данных, более широкие (имеющие большую разрядность) шины данных и режим групповых пересылок (пересылки нескольких слов) дают увеличение производительности за счет увеличения стоимости.

Главное устройство шины - это устройство, которое может инициировать транзакции чтения или записи. ЦП, например, всегда является главным устройством шины. Шина имеет несколько главных устройств, если имеется несколько ЦП или когда устройства ввода/вывода могут инициировать транзакции на шине. Если имеется несколько таких устройств, то требуется схема арбитража, чтобы решить, кто следующий захватит шину. Арбитраж часто основан либо на схеме с фиксированным приоритетом, либо на более "справедливой" схеме, которая случайным образом выбирает, какое главное устройство захватит шину.

В настоящее время используются два типа шин, отличающиеся способом коммутации: шины с коммутацией цепей (circuit-switched bus) и шины с коммутацией пакетов (packet-switched bus), получившие свои названия по аналогии со способами коммутации в сетях передачи данных. Шина с коммутацией пакетов при наличии нескольких главных устройств шины обеспечивает значительно большую пропускную способность по сравнению с шиной с коммутацией цепей за счет разделения транзакции на две логические части: запроса шины и ответа. Такая методика получила название "расщепления" транзакций (split transaction). (В некоторых системах такая возможность называется шиной соединения/разъединения (connect/disconnect) или конвейерной шиной (pipelined bus). Транзакция чтения разбивается на транзакцию запроса чтения, которая содержит адрес, и транзакцию ответа памяти, которая содержит данные. Каждая транзакция теперь должна быть помечена (тегирована) соответствующим образом, чтобы ЦП и память могли сообщить что есть что.

Шина с коммутацией цепей не делает расщепления транзакций, любая транзакция на ней есть неделимая операция. Главное устройство запрашивает шину, после арбитража помещает на нее адрес и блокирует шину до окончания обслуживания запроса. Большая часть этого времени обслуживания при этом тратится не на выполнение операций на шине (например, на задержку выборки из памяти). Таким образом, в шинах с коммутацией цепей это время просто теряется. Расщепленные транзакции делают шину доступной для других главных устройств пока память читает слово по запрошенному адресу. Это, правда, также означает, что ЦП должен бороться за шину для посылки данных, а память должна бороться за шину, чтобы вернуть данные. Таким образом, шина с расщеплением транзакций имеет более высокую пропускную способность, но обычно она имеет и большую задержку, чем шина, которая захватывается на все время выполнения транзакции. Транзакция называется расщепленной, поскольку произвольное количество других пакетов или транзакций могут использовать шину между запросом и ответом.

Последний вопрос связан с выбором типа синхронизации и определяет является ли шина синхронной или асинхронной. Если шина синхронная, то она включает сигналы синхронизации, которые передаются по линиям управления шины, и фиксированный протокол, определяющий расположение сигналов

адреса и данных относительно сигналов синхронизации. Поскольку практически никакой дополнительной логики не требуется для того, чтобы решить, что делать в следующий момент времени, эти шины могут быть и быстрыми, и дешевыми. Однако они имеют два главных недостатка. Все на шине должно происходить с одной и той же частотой синхронизации, поэтому из-за проблемы перекоса синхросигналов, синхронные шины не могут быть длинными. Обычно шины процессор-память синхронные.

Асинхронная шина, с другой стороны, не тактируется. Вместо этого обычно используется старт-стопный режим передачи и протокол "рукопожатия" (handshaking) между источником и приемником данных на шине. Эта схема позволяет гораздо проще приспособить широкое разнообразие устройств и удлинить шину без беспокойства о перекосе сигналов синхронизации и о системе синхронизации. Если может использоваться синхронная шина, то она обычно быстрее, чем асинхронная, из-за отсутствия накладных расходов на синхронизацию шины для каждой транзакции. Выбор типа шины (синхронной или асинхронной) определяет не только пропускную способность, но также непосредственно влияет на емкость системы ввода/вывода в терминах физического расстояния и количества устройств, которые могут быть подсоединены к шине. Асинхронные шины по мере изменения технологии лучше масштабируются. Шины ввода/вывода обычно асинхронные.

Обычно количество и типы устройств ввода/вывода в вычислительных системах не фиксируются, что позволяет пользователю самому подобрать необходимую конфигурацию. Шина ввода/вывода компьютера может рассматриваться как шина расширения, обеспечивающая постепенное наращивание устройств ввода/вывода. Поэтому стандарты играют огромную роль, позволяя разработчикам компьютеров и устройств ввода/вывода работать независимо. Появление стандартов определяется разными обстоятельствами.

Иногда широкое распространение и популярность конкретных машин становятся причиной того, что их шина ввода/вывода становится стандартом де факто. Примерами таких шин могут служить PDP-11 Unibus и IBM PC-AT Bus. Иногда стандарты появляются также в результате определенных достижений по стандартизации в некотором секторе рынка устройств ввода/вывода. Интеллектуальный периферийный интерфейс (IPI - Intelligent Peripheral Interface) и Ethernet являются примерами стандартов, появившихся в результате кооперации производителей. Успех того или иного стандарта в значительной степени определяется его принятием такими организациями как ANSI (Национальный институт по стандартизации США) или IEEE (Институт инженеров по электротехнике и радиоэлектронике). Иногда стандарт шины может быть прямо разработан одним из комитетов по стандартизации: примером такого стандарта шины является FutureBus.

Одной из популярных шин персональных компьютеров была системная шина, XT- Bus - шина архитектуры XT - первая в семействе IBM PC.

Относительно проста, поддерживает обмен 8-разрядными данными внутри 20-разрядного (1 Мб) адресного пространства (обозначается как "разрядность 8/20"), работает на частоте 4.77 МГц. Совместное использование линий IRQ в общем случае невозможно. Конструктивно оформлена в 62-контактних разъемах.

ISA (Industry Standard Architecture - архитектура промышленного стандарта) - основная шина на компьютерах типа PC AT (другое название - AT-Bus). Является расширением XT-Bus, разрядность - 16/24 (16 Мб), тактовая частота - 8 МГц, предельная пропускная способность -5.55 Мб/с. Разделение IRQ также невозможно. Возможна нестандартная организация Bus Mastering, но для этого нужен запрограммированный 16-разрядный канал DMA. Конструктивно выполнено в виде 62-контактного разъема XT-Bus с прилегающим к нему 36-контактным разъемом расширения.

EISA (Enhanced ISA - расширенная ISA) - функциональное и конструктивное расширение ISA. Внешне разъемы имеют такой же вид, как и ISA, и в них могут вставляться платы ISA, но в глубине разъема находятся дополнительные ряды контактов EISA, а платы EISA имеют более высокую ножевую часть разъема с дополнительными рядами контактов. Разрядность - 32/32 (адресное пространство - 4 Гб), работает также на частоте 8 МГц. Предельная пропускная способность - 32 Мб/с. Поддерживает Bus Mastering - режим управления шиной со стороны любого из устройств на шине, имеет систему арбитража для управления доступом устройств у шине, позволяет автоматически настраивать параметры устройств, возможно разделение каналов IRQ и DMA.

Bus Mastering - способностъ внешнего устройства самостоятельно, без участия процессора, управлять шиной (пересылать данные, выдавать команды и сигналы управления). На время обмена устройство захватывает шину и становится главным, или ведущим (master) устройством. Такой подход обычно используется для освобождения процессора от операций пересылки команд и/или данных между двумя устройствами на одной нише. Частным случаем Bus Mastering является режим DMA, который осуществляет только внепроцессорную пересылку данных; в классической архитектуре PC этим занимается контроллер DMA, общий для всех устройств. Каждое же Bus Mastering-устройство имеет собственный подобный контроллер, что позволяет избавиться от проблем с распределением DMA- каналов и преодолеть ограничения стандартного DMA- контроллера (16- разрядность, способность адресовать только первые 16 Мб ОЗУ, низкое быстродействие и т.п.).

MCA (Micro (Channel Architecture - микроканальная архитектура) - шинакомпьютеров PS/2 фирмы IBM. Не совместима ни с одной другой, разрядность - 32/32, (базовая - 8/24, остальные - в качестве расширений). Поддерживает Bus Mastering, имеет арбитраж и автоматическую конфигурацию, синхронная (жестко фиксирована длительность цикла обмена), предельная пропускная способность - 40 Мб/с. Конструктивно выглядит, как

одно- трехсекционный разъем (такой же, как у VLB). Первая, основная, секция - 8-разрядная (90 контактов), вторая - 16- разрядное расширение (22 контакта), третья - 32- разрядное расширение (52 контакта). В основной секции предусмотрены линии для передачи звуковых сигналов. Дополнительно рядом с одним из разъемов может устанавливаться разъем видеорасширения (20 контактов). EISA и MCA во многом параллельны, появление EISA было обусловлено собственностью IBM на архнтектуру MCA.

VLB (VESA Local Bus - локальная шина стандарта VESA) - 32-разрядное (дополнение к шине ISA. Конструктивно представляет собой дополнительный разъем (116- контактный, как у MCA) при разъеме ISA. Разрядность - 32/32, тактовая частота - 25..50 МГц, предельная скорость обмена - 130 Мб/с. Электрически выполнена в виде расширения локальной шины процессора - большинство входных и выходных сигналов процессора передаются непосредственно VLB-платам без промежуточной буферизации. Из- за этого возрастает нагрузка на выходные каскады процессора, ухудшается качество сигналов на локальной шине и снижается надежность обмена по ней. Поэтому VLB имеет жесткое ограничение на количество устанавливаемых устройств: при 33 МГц - три, 40 МГц - два, и при 50 МГц - одно, причем желательно - интегрированное в системную плату.

PCI (Peripheral Component Interconnect - соединение внешних компонент) - развитие VLB в сторону EISA/MCA. Не совместима ни с какими другими, разрядность - 32/32 (расширенный вариант - 64/64), тактовая частота -до 33 МГц (PCI 2.1 - до 66 МГц), пропускная способность - до 132 Мб/с (264 Мб/с для 32/32 на 66 МГц и 528 Мб/с для 64/64 на 66 МГц), поддержка Bus Mastering и автоконфигурации. Количество разъемов шины на одном сегменте ограничено четырьмя. Сегментов может быть несколько, они соединяются друг с другом посредством мостов (bridge). Сегменты могут объединяться в различные топологии (дерево, звезда и т.п.). Самая популярная шина в настоящее время, используется также на других компьютерах. Разъем похож на MCA/VLB, но чуть длиннее (124 контакта). 64-разрядный разъем имеет дополнительную 64-контактную секцию с собственным ключом. Все разъемы и карты к ним делятся на поддерживающие уровни сигналов 5В, 3.3 В и универсальные; первые два типа должны соответствовать друг другу, универсальные карты ставятся в любой разъем.

Существует также расширение MediaBus, введенное фирмой ASUSTek - дополнительный разъем содержит сигналы шины ISA.

PCMCIA (Personal Computer Memory Card International Association - ассоциация производителей плат памяти для персональных компьютеров) - внешняя шина компьютеров класса NoteBook. Другое название модуля PCMCIA - PC Card. Предельно проста, разрядность - 16/26 (адресное пространство - 64 Мб), поддерживает автоконфигурацию, возможно подключение и отключение устройств в процессе работы компьютера. Конструктив - миниатюрный 68-контактный разъем. Контакты питания

сделаны более длинными, что позволяет вставлять и вынимать карту при включенном питании компьютера.

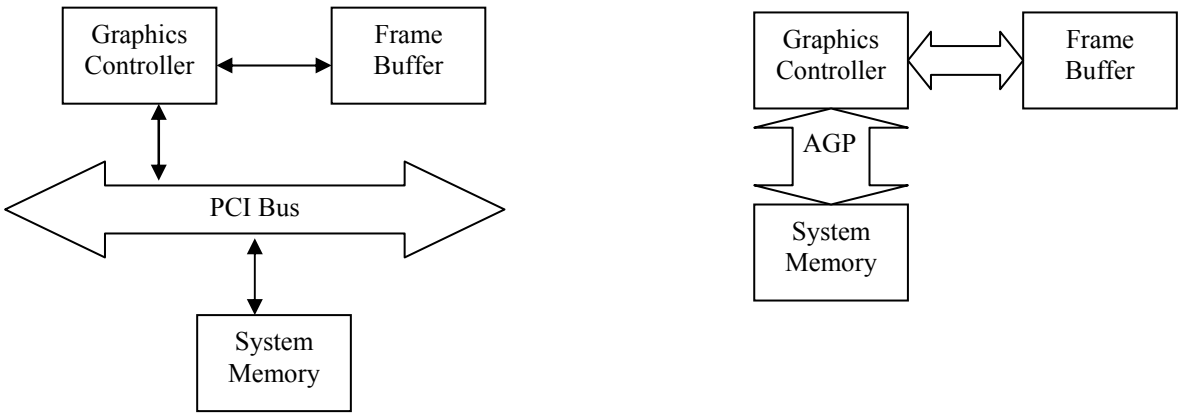
7.3. Шины ввода/вывода

7.3.1. Шина AGP

Шина персонального компьютера (PC) постоянно терпит множество изменений в связи с повышаемыми к ней требованиями.

Так ускоренный графический порт (AGP) это расширение шины PCI, чье назначение обработка больших массивов данных 3D графики. Intel разрабатывала AGP, для решения двух проблем перед внедрением 3D графики на PCI. Во-первых, 3D графика требуется как можно больше памяти информации текстурных карт (texture maps) и z-буфера (z- buffer). Чем больше текстурных карт доступно для 3D приложений, тем лучше выглядит конечный результат. При нормальных обстоятельствах z- буфер, который содержит информацию относящуюся к представлению глубины изображения, использует ту же память как и текстуры. Этот конфликт передоставляет разработчикам 3D множество вариантов для выбора оптимального решения, которое они привязывают к большой значимости памяти для текстур и z-буфера, и результаты напрямую влияют на качество выводимого изображения.

Схемы PCI и AGP



а) PCI структура **б)AGP структура**

Рис. 7.1.

Разработчики PC имели ранее возможность использовать системную память для хранения информации о текстурах и z-буфера, но ограничение в таком подходе, была передача такой информации через шину PCI. Производительность графической подсистемы и системной памяти ограничиваются физическими характеристиками шины PCI (рис.7.1, а). Кроме того, ширина полосы пропускания PCI, или ее емкость, не достаточна для

обработки графики в режиме реального времени. Чтобы решить эти проблемы Intel разработала AGP.

Если определить кратко, что такое AGP, то это - прямое соединение между графической подсистемой и системной памятью (рис.7.1, б). Это решение позволяет обеспечить значительно лучшие показатели передачи данных, чем при передаче через шину PCI, и явно разрабатывалось, чтобы удовлетворить требованиям вывода 3D графики в режиме реального времени. AGP позволит более эффективно использовать память страничного буфера (frame buffer), тем самым увеличивая производительность 2D графики также, как увеличивая скорость прохождения потока данных 3D графики через систему.

Определение AGP, как вид прямого соединения между графической подсистемой и системной памятью, называется соединение point-to-point. В действительности, AGP соединяет графическую подсистему с блоком управления системной памятью, разделяя этот доступ к памяти с центральным процессором компьютера (CPU).

7.3.2. Шина USB

Шина USB предназначена для обеспечения обмена данными между компьютером (центральным процессором устройства) и подсоединенными к нему периферийными устройствами (ПУ) в условиях динамического (горячего) изменения конфигурации системы.

При проектировании новой шины особое внимание обращалось на следующие показатели:

• простоту изменения конфигурации системы;

• стоимость законченного решения при пропускной способности до 12 Мбит/с;

• возможность передачи потоков аудио- и сжатых видеоданных в реальном времени;

• обеспечение одновременной передачи разных типов данных;

• адаптацию к существующей инфраструктуре ПК и возможность быстрого включения интерфейса шины в представленное на рынке прикладное ПО;

• стимулирование разработки новых классов устройств, расширяющих возможности ПК.

Возможности USB следуют из ее технических характеристик:

• высокая скорость обмена (full-speed signaling bit rate) -12 Mb/s;

• максимальная длина кабеля для высокой скорости обмена - 5 m;

• низкая скорость обмена (low-speed signaling bit rate) - 1.5 Mb/s;

• максимальная длина кабеля для низкой скорости обмена -3т;

• максимальное количество подключенных устройств (включая размножители) - 127;

• возможно подключение устройств с различными скоростями обмена;

• отсутствие необходимости в установке пользователем дополнительных элементов, таких как терминаторы для SCSI;

• напряжение питания для периферийных устройств - 5 V;

• максимальный ток потребления на одно устройство - 500 mA (это не означает, что через USB можно запитать устройства с общим током потребления 127 x 500 mA=63.5 A).

Поэтому целесообразно подключать к USB практически любые периферийные устройства, кроме цифровых видеокамер и жестких высокоскоростных дисков. Особенно удобен этот интерфейс для подключения часто подключаемых отключаемых приборов, таких как цифровые фотокамеры. Конструкция разъемов для USB рассчитана на многократное сочленение - расчленение. Возможность использования только двух скоростей обмена данными ограничивает применяемость шины, но существенно уменьшает количество линий интерфейса и упрощает аппаратную реализацию. Питание непосредственно от USB возможно только для устройств с малым потреблением, таких как клавиатуры, мыши, джостики и т. д.

Сама шина - это многоуровневая иерархическая система. На физическом уровне топология шины представляет собой корневидную структуру (рис. 7.2) - многоуровневую звезду (в терминологии стандарта), при которой соединения могут формировать цепочки и звезды. Закольцовка соединений в системе не допускается (этому, в частности, препятствует разная конструкция разъемов входного и выходного портов шины USB).

В самой верхней части "корня" шины USB находится корневой концентратор, обеспечивающий связь периферии с компьютером (хостом). В текущей реализации стандарта допускается наличие только одного корневого концентратора, хотя и делается очень важная оговорка о возможности модернизации в будущем с целью поддержки нескольких корневых узлов в одной системе, что позволит, по мнению автора, создавать своего рода микросети в пределах, например, одного помещения. Уже практически готовы основные, системообразующие, решения и компоненты: концентраторы, мультиплексированная шина, программные и аппаратные средства ее поддержки.

Центром каждой звезды является узел (концентратор) шины USB, который обеспечивает набор двухточечных соединений с другими узлами и/или функциями, лежащими вниз по потоку (т. е. на большем удалении от компьютера). Узел состоит из двух функциональных элементов - повторителя, служащего для управления коммутацией потоков информации между входным и выходными портами узла, и контроллера, предназначенного для управления статусом (состоянием) узла и его портов (рис. 7.2).

Топология шины USB

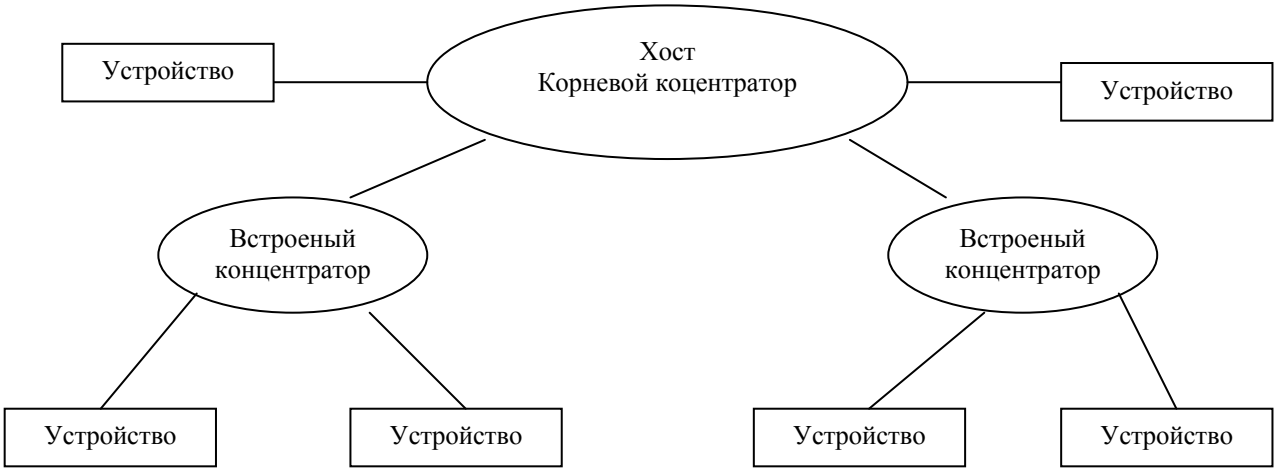


Рис. 7.2.

Функции шины USB - это возможности, которые формируют полезные свойства сети USB, например выход в линию ISDN, получение данных от джойстика или вывод сигнала на звуковые колонки. Понятие функции не эквивалентно определению устройства, поскольку последнее может реализовать сразу несколько функций. В этом случае оно называется составным и рассматривается системой как узел с несколькими постоянно подключенными однофункциональными устройствами. Подобное устройство должно содержать встроенный концентратор шины USB.

Все устройства и узлы шины USB могут иметь собственные источники питания или запитываться от вышестоящего узла USB. (Запитываемый от шины узел обеспечивает работу до четырех ПУ, а с собственным питанием - до семи и более.) Все эти устройства в соответствии со стандартом шины подразделяются на классы, которые образуют свою иерархию. Классами-родоначальниками являются узлы и функции. Введение классов устройств должно, по мнению создателей шины, способствовать стандартизации аналогичных по назначению периферийных устройств разных производителей. По мере необходимости разработчики смогут, кроме стандартных классов, определять новые классы ПУ.

Интерфейс между USB и компьютером называется хост-контроллером (host controller, HC), который реализуется комбинацией аппаратных средств и драйвера хост-контроллера (host controller driver, HCD). Хост-контроллер отвечает за выполнение следующих операций:

• обнаружение подключения/отключения устройств;

• потоки управляющей информации между хостом и ПУ;

• управление потоками данных по шине, в частности выполнение протокола шины;

• сбор информации о статусе и активности ПУ-системы, а также формирование отчетов о состоянии системы USB;

• выделение ПУ определенных лимитов энергоресурсов системы (особенно актуально

для мобильных систем).

Хост-контроллер USB активно взаимодействует с различными службами операционной системы. Например, при наличии в ОС службы управления энергоресурсами APM (advanced power management) ПО USB перехватывает и выполняет запросы этой службы на приостановку функционирования и восстановление рабочего состояния конкретных устройств.

Прикладной интерфейс USB содержит драйверы стандартных классов устройств для данной ОС. Здесь используются обращения к специфическим службам ОС, в частности PnP для Windows 95. Разработчики нестандартного оборудования должны включать свои драйверы в этот уровень ПО шины USB.

Еще один важный компонент верхнего уровня ПО шины - система конфигурирования шины и идентификации ПУ, поставляемая разработчиком ОС или независимыми производителями ПО. Эта система управляет всеми узлами сети, в том числе корневым концентратором, и является частью службы управления энергопотреблением компьютерной системы.

Ключевым элементом ПО USB является драйвер USBD, поставляемый, как сказано в стандарте шины, разработчиком ОС. На него ложится вся диспетчеризация активности на шине. Драйвер транслирует запросы ввода/вывода клиентского ПО в вызовы HCD. Например, USBD на основании данных запроса на подключение нового ПУ (число конечных точек в устройстве, допустимые типы и объемы передач данных и т. д.) дает отказ или удовлетворяет запрос, исходя из свободных ресурсов шины.

USBD опирается на драйвер хост-контроллера, скрывающий особенности аппаратных решений USB от вышележащего ПО. Драйвер хост-контроллера отслеживает выполнение текущих запросов на доступ к шине и обеспечивает бездефицитное выделение имеющихся ресурсов шины. Драйвер хост-контроллера также поставляется разработчиком ОС и содержит в настоящее время два аппаратных интерфейса: UHCI (universal host controller interface) и OHCI (open host controller interface).

Как и в любой сложной многоуровневой системе, использующей общий коммуникационный канал, передача потоков информации между хостом и ПУ по шине USB требует взаимодействия многих программных и аппаратных компонентов, каждый из которых имеет свою сферу ответственности. Это придает особое значение протоколу обмена между элементами системы.

В шине USB используется мультиплексирование передаваемых данных с временным уплотнением (time division multiplexing, TDM). Основу логической модели передачи данных составляют пакеты. Размер пакета переменный, он зависит от многих факторов. Хост-контроллер объединяет пакеты в кадры, длительность которых 0,001 с. Порядок следования пакетов в кадре

определяется драйвером хост-контроллера, однако для каждого получателя информации (логического канала передачи данных) гарантируется сохранение последовательности поступления данных.

Системное ПО шины и специальные протоколы обмена скрывают от клиентского ПО (прикладных программ) сложность централизованного управления маркерным доступом к совместно используемым ресурсам шины USB, сводя его к системе двухточечных связей. Этим USB отличается от таких шин, как PCI, EISA, PCMCIA, где клиентское ПО напрямую работает с адресатом.

7.3.3. Шины IDE и SCSI

Одними из наиболее популярных шин ввода-вывода в настоящее время являются шины IDE и SCSI.

Под термином IDE (Integrated Drive Electronics - электроника, встроенная в привод)), или ATA (AT Attachment - подключаемый к AT) понимается простой и недорогой интерфейс для PC AT. Все функции по управлению накопителем обеспечивает встроенный контроллер, а 40-проводной соединительный кабель является фактически упрощенным сегментом 16-разрядной магистрали AT-Bus (ISA). Простейший адаптер IDE содержит только адресный дешифратор - все остальные сигналы заводятся прямо на разъем ISA. Адаптеры IDE обычно не содержат собственного BIOS - все функции поддержки IDE встроены в системный BIOS PC AT. Однако интеллектуальные или кэширующие контроллеры могут иметь собственный BIOS, подменяющий часть или все функции системного.

Основной режим работы устройств IDE - программный обмен (PIO) под управлением центрального процессора, однако все современные винчестеры EIDE поддерживают обмен в режиме DMA, а большинство контроллеров - режим Bus Mastering.

Под термином SCSI - Small Computer System Interface (Интерфейс малых вычислительных систем) обычно понимается набор стандартов, разработанных Национальным институтом стандартов США (ANSI) и определяющих механизм реализации магистрали передачи данных между системной шиной компьютера и периферийными устройствами. На сегодняшний день приняты два стандарта (SCSI-1 и SCSI-2). Стандарт SCSI-3 находится в процессе доработки.

Начальный стандарт 1986 года, известный теперь под названием SCSI-1, определял рабочие спецификации протокола шины, набор команд и электрические параметры. В 1992 году этот стандарт был пересмотрен с целью устранения недостатков первоначальной спецификации (особенно в части синхронного режима передачи данных) и добавления новых возможностей повышения производительности, таких как "быстрый режим" (fast mode), "широкий режим" (wide mode) и помеченные очереди. Этот пересмотренный

стандарт получил название SCSI-2 и в настоящее время используется большинством поставщиков вычислительных систем.

Первоначально SCSI предназначался для использования в небольших дешевых системах и поэтому был ориентирован на достижение хороших результатов при низкой стоимости. Характерной его чертой является простота, особенно в части обеспечения гибкости конфигурирования периферийных устройств без изменения организации основного процессора. Главной особенностью подсистемы SCSI является размещение в периферийном оборудовании интеллектуального контроллера.

Для достижения требуемого высокого уровня независимости от типов периферийных устройств в операционной системе основной машины, устройства SCSI представляются имеющими очень простую архитектуру. Например, геометрия дискового накопителя представляется в виде линейной последовательности одинаковых блоков, хотя в действительности любой диск имеет более сложную многомерную геометрию, содержащую поверхности, цилиндры, дорожки, характеристики плотности, таблицу дефектных блоков и множество других деталей. В этом случае само устройство или его контроллер несут ответственность за преобразование упрощенной SCSI модели в данные для реального устройства.

Стандарт SCSI-2 определяет в частности различные режимы: Wide SCSI, Fast SCSI и Fast-and-Wide SCSI. Стандарт SCSI-1 определяет построение периферийной шины на основе 50-жильного экранированного кабеля, описывает методы адресации и электрические характеристики сигналов. Шина данных SCSI-1 имеет разрядность 8 бит, а максимальная скорость передачи составляет 5 Мбайт/сек. Fast SCSI сохраняет 8-битовую шину данных и тем самым может использовать те же самые физические кабели, что и SCSI-1. Он отличается только тем, что допускает передачи со скоростью 10 Мбайт/сек в синхронном режиме. Wide SCSI удваивает либо учетверяет разрядность шины данных (либо 16, либо 32 бит), допуская соответственно передачи со скоростью либо 10, либо 20 Мбайт/сек. В комбинации Fast-and-Wide SCSI возможно достижение скоростей передачи 20 и 40 Мбайт/сек соответственно.

Однако поскольку в обычном 50-жильном кабеле просто не хватает жил, комитет SCSI решил расширить спецификацию вторым 66-жильным кабелем (так называемый B-кабель). B-кабель имеет дополнительные линии данных и ряд других сигнальных линий, позволяющие реализовать режим Fast-and-Wide.

В реализации режима Wide SCSI предложена также расширенная адресация, допускающая подсоединение к шине до 16 устройств (вместо стандартных восьми). Это значительно увеличивает гибкость подсистемы SCSI, правда приводит к появлению дополнительных проблем, связанных с эффективностью ее использования.

Реализация режимов Wide-SCSI и Fast-and-Wide SCSI до 1994 года редко использовалась, поскольку эффективность их применения не была достаточно высокой. Однако широкое распространение дисковых массивов и дисковых

накопителей со скоростью вращения 7200 оборотов в минуту делают эту технологию весьма актуальной.

8. Периферийные устройства

Как правило, периферийные устройства компьютеров делятся на устройства ввода, устройства вывода и внешние запоминающие устройства (осуществляющие как ввод данных в машину, так и вывод данных из компьютера). Основной обобщающей характеристикой устройств ввода/вывода может служить скорость передачи данных (максимальная скорость, с которой данные могут передаваться между устройством ввода/вывода и основной памятью или процессором). В табл. 8.1. представлены основные устройства ввода/вывода, применяемые в современных компьютерах, а также указаны примерные скорости обмена данными, обеспечиваемые этими устройствами.

Таблица 8.1.

Примеры устройств ввода/вывода

Тип устройства	Направление передачи данных	Скорость передачи данных (Кбайт/с)
Клавиатура	Ввод	0.01
Мышь	Ввод	0.02
Голосовой ввод	Ввод	0.02
Сканер	Ввод	200.0
Голосовой вывод	Вывод	0.06
Строчный принтер	Вывод	1.00
Лазерный принтер	Вывод	100.00
Графический дисплей	Вывод	30000.00
(ЦП (r) буфер кадра)	Вывод	200.0
Оптический диск	ЗУ	500.00
Магнитная лента	ЗУ	2000.00
Магнитный диск	ЗУ	2000.00

Мы рассмотрим наиболее быстрые из этих устройств: магнитные и магнитооптические диски.

8.1. Магнитные и магнитооптические диски

Сначала рассмотрим основную терминологию, применяемую при описании магнитных дисков и контроллеров, а затем приведем типовые характеристики нескольких современных дисковых подсистем.

Дисковый накопитель обычно состоит из набора пластин, представляющих собой металлические диски, покрытые магнитным материалом и соединенные между собой при помощи центрального шпинделя. Для записи данных используются обе поверхности пластины. В современных дисковых накопителях используется от 4 до 9 пластин. Шпиндель вращается с высокой постоянной скоростью (обычно 3600, 5400 или 7200 оборотов в минуту). Каждая пластина содержит набор концентрических записываемых

дорожек. Обычно дорожки делятся на блоки данных объемом 512 байт, иногда называемые секторами. Количество блоков, записываемых на одну дорожку зависит от физических размеров пластины и плотности записи.

Данные записываются или считываются с пластин с помощью головок записи/считывания, по одной на каждую поверхность. Линейный двигатель представляет собой электро-механическое устройство, которое позиционирует головку над заданной дорожкой. Обычно головки крепятся на кронштейнах, которые приводятся в движение каретками. Цилиндр - это набор дорожек, соответствующих одному положению каретки. Накопитель на магнитных дисках (НМД) представляет собой набор пластин, магнитных головок, кареток, линейных двигателей плюс воздухонепроницаемый корпус. Дисковым устройством называется НМД с относящимися к нему электронными схемами.

Производительность диска является функцией времени обслуживания, которое включает в себя три основных компонента: время доступа, время ожидания и время передачи данных. Время доступа - это время, необходимое для позиционирования головок на соответствующую дорожку, содержащую искомые данные. Оно является функцией затрат на начальные действия по ускорению головки диска (порядка 6 мс), а также функцией числа дорожек, которые необходимо пересечь на пути к искомой дорожке. Характерные средние времена поиска - время, необходимое для перемещения головки между двумя случайно выбранными дорожками, лежат в диапазоне 10-20 мс. Время перехода с дорожки на дорожку меньше 10 мс и обычно составляет 2 мс.

Вторым компонентом времени обслуживания является время ожидания. Чтобы искомый сектор повернулся до совмещения с положением головки требуется некоторое время. После этого данные могут быть записаны или считаны. Для современных дисков время полного оборота лежит в диапазоне 8-16 мс, а среднее время ожидания составляет 4-8 мс.

Последним компонентом является время передачи данных, т.е. время, необходимое для физической передачи байтов. Время передачи данных является функцией от числа передаваемых байтов (размера блока), скорости вращения, плотности записи на дорожке и скорости электроники. Типичная скорость передачи равна 1-4 Мбайт/с.

В состав компьютеров часто входят специальные устройства, называемые дисковыми контроллерами. К каждому дисковому контроллеру может подключаться несколько дисковых накопителей. Между дисковым контроллером и основной памятью может быть целая иерархия контроллеров и магистралей данных, сложность которой определяется главным образом стоимостью компьютера. Поскольку время передачи часто составляет очень небольшую часть общего времени доступа к диску, контроллер в высокопроизводительной системе разъединяет магистрали данных от диска на время позиционирования так, что другие диски, подсоединенные к контроллеру, могут передавать свои данные в основную память. Поэтому время

доступа к диску может увеличиваться на время, связанное с накладными расходами контроллера на организацию операции ввода/вывода.

Рассмотрим теперь основные составляющие времени доступа к диску в типичной подсистеме SCSI. Такая подсистема включает в себя четыре основных компонента: основной компьютер, главный адаптер SCSI, встроенный в дисковое устройство контроллер и собственно накопитель на магнитных дисках. Когда операционная система получает запрос от пользователя на выполнение операции ввода/вывода, она превращает этот запрос в набор команд SCSI. Запрашивающий процесс при этом блокируется и откладывается до завершения операции ввода/вывода (если только это был не запрос асинхронной передачи данных). Затем команды пересылаются по системе шин в главный адаптер SCSI, к которому подключен необходимый дисковый накопитель. После этого ответственность за выполнение взаимодействия с целевыми контроллерами и их устройствами ложится на главный адаптер.

Затем главный адаптер выбирает целевое устройство, устанавливая сигнал на линии управления шины SCSI (эта операция называется фазой выбора). Естественно, шина SCSI должна быть доступна для этой операции. Если целевое устройство возвращает ответ, то главный адаптер пересылает ему команду (это называется фазой команды). Если целевой контроллер может выполнить команду немедленно, то он пересылает в главный адаптер запрошенные данные или состояние. Команда может быть обслужена немедленно, только если это запрос состояния, или команда запрашивает данные, которые уже находятся в кэш-памяти целевого контроллера. Обычно же данные не доступны, и целевой контроллер выполняет разъединение, освобождая шину SCSI для других операций. Если выполняется операция записи, то за фазой команды на шине немедленно следует фаза данных, и данные помещаются в кэш-память целевого контроллера. Подтверждение записи обычно не происходит до тех пор, пока данные действительно не запишутся на поверхность диска.

После разъединения, целевой контроллер продолжает свою собственную работу. Если в нем не предусмотрены возможности буферизации команд (создание очереди команд), ему надо только выполнить одну команду. Однако, если создание очереди команд разрешено, то команда планируется в очереди работ целевого контроллера, при этом обрабатывается команда, обладающая наивысшим приоритетом в очереди. Когда запрос станет обладать наивысшим приоритетом, целевой контроллер должен вычислить физический адрес (или адреса), необходимый для обслуживания операции ввода/вывода. После этого становится доступным дисковый механизм: позиционируется каретка, подготавливается соответствующая головка записи/считывания и вычисляется момент появления данных под головкой. Наконец, данные физически считываются или записываются на дорожку. Считанные данные запоминаются

в кэш-памяти целевого контроллера. Иногда целевой контроллер может выполнить считывание с просмотром вперед.

После завершения операции ввода/вывода целевой контроллер в случае свободы шины соединяется с главным адаптером, вслед за чем выполняется фаза данных (при передаче данных из целевого контроллера в главный адаптер) и фаза состояния для указания результата операции. Когда главный адаптер получает фазу состояния, он проверяет корректность завершения физической операции в целевом контроллере и соответствующим образом информирует операционную систему.

Одной из характеристик процесса ввода/вывода SCSI является большое количество шагов, которые обычно не видны пользователю. Обычно на шине SCSI происходит смена семи фаз (выбор, команда, разъединение, повторное соединение, данные, состояние, разъединение). Естественно каждая фаза выполняется за некоторое время, расходуемое на использование шины. Многие целевые контроллеры (особенно медленные устройства подобные магнитным лентам и компакт-дискам) потребляют значительную часть времени на реализацию фаз выбора, разъединения и повторного соединения.

Варианты применения высокопроизводительных подсистем ввода/вывода широко варьируются в зависимости от требований, которые к ним предъявляются. Они охватывают диапазон от обработки малого числа больших массивов данных, которые необходимо реализовать с минимальной задержкой (ввод/вывод суперкомпьютера), до большого числа простых заданий, которые оперируют с малыми объемами данных (обработка транзакций).

Запросы на ввод/вывод заданной рабочей нагрузки можно характеризовать в терминах трех метрик: производительность, время ожидания и пропускная способность. Производительность определяется числом запросов на обслуживание, получаемых в единицу времени. Время ожидания определяет время, необходимое на обслуживание индивидуального запроса. Пропускная способность определяет количество данных, передаваемых между устройствами, требующими обслуживания, и устройствами, выполняющими обслуживание.

Ввод/вывод суперкомпьютера почти полностью определяется последовательным механизмом. Обычно данные передаются с диска в память большими блоками, а результаты записываются обратно на диск. В таких применениях требуется высокая пропускная способность и минимальное время ожидания, однако они характеризуются низкой производительностью. В отличие от этого обработка транзакций характеризуется огромным числом случайных обращений, относительно небольшими отрезками работы и требует умеренного времени ожидания при очень высокой производительности.

Так как системы обработки транзакций тратят большую часть времени обслуживания на поиск и ожидание, технологические успехи, приводящие к сокращению времени передачи, не будут оказывать особого влияния на производительность таких систем. С другой стороны, в научных применениях

на поиск данных и на их передачу затрачивается одинаковое время, и поэтому производительность таких систем оказывается очень чувствительной к любым усовершенствованиям в технологии изготовления дисков. Как будет показано ниже, можно организовать матрицу дисков таким образом, что будет обеспечена высокая производительность ввода/вывода для широкого спектра рабочих нагрузок.

В последние годы плотность записи на жестких магнитных дисках увеличивается на 60% в год при ежеквартальном снижении стоимости хранения одного Мегабайта на 12%. По данным фирмы Dataquest такая тенденция сохранится и в ближайшие два года. Сейчас на рынке представлен широкий ассортимент дисковых накопителей емкостью до 9.1 Гбайт. При этом среднее время доступа у самых быстрых моделей достигает 8 мс. Например, жесткий диск компании Seagate Technology имеет емкость 4.1 Гбайт и среднее время доступа 8 мс при скорости вращения 7200 оборот/мин. Улучшаются также характеристики дисковых контроллеров на базе новых стандартов Fast SCSI-2 и Enhanced IDE. Предполагается увеличение скорости передачи данных до 13 Мбайт/с. Надежность жестких дисков также постоянно улучшается. Например, некоторые модели дисков компаний Conner Peripherals Inc., Micropolis Corp. и Hewlett-Packard имеют время наработки на отказ от 500 тысяч до 1 миллиона часов

Другим направлением развития систем хранения информации являются магнитооптические диски. Запись на магнитооптические диски (МО-диски) выполняется при взаимодействии лазера и магнитной головки. Луч лазера разогревает до точки Кюри (температуры потери материалом магнитных свойств) микроскопическую область записывающего слоя, которая при выходе из зоны действия лазера остывает, фиксируя магнитное поле, наведенное магнитной головкой. В результате данные, записанные на диск, не боятся сильных магнитных полей и колебаний температуры. Все функциональные свойства дисков сохраняются в диапазоне температур от -20 до +50 градусов Цельсия.

МО-диски уступают обычным жестким магнитным дискам лишь по времени доступа к данным. Предельное достигнутое МО-дисками время доступа составляет 19 мс. Магнитооптический принцип записи требует предварительного стирания данных перед записью, и соответственно, дополнительного оборота МО-диска. Однако завершенные недавно исследования в SONY и IBM показали, что это ограничение можно устранить, а плотность записи на МО-дисках можно увеличить в несколько раз. Во всех других отношениях МО-диски превосходят жесткие магнитные диски.

В магнитооптическом дисководе используются сменные диски, что обеспечивает практически неограниченную емкость. Стоимость хранения единицы данных на МО-дисках в несколько раз меньше стоимости хранения того же объема данных на жестких магнитных дисках.

Сегодня на рынке МО-дисков предлагается более 150 моделей различных фирм. Одно из лидирующих положений на этом рынке занимает компания Pinnacle Micro Inc. Для примера, ее дисковод Sierra 1.3 Гбайт обеспечивает среднее время доступа 19 мс и среднее время наработки на отказ 80000 часов. Для серверов локальных сетей и рабочих станций компания Pinnacle Micro предлагает целый спектр многодисковых систем емкостью 20, 40, 120, 186 Гбайт и даже 4 Тбайт. Для систем высокой готовности Pinnacle Micro выпускает дисковый массив Array Optical Disk System, который обеспечивает эффективное время доступа к данным не более 11 мс при скорости передачи данных до 10 Мбайт/с.

9. Организация RISC системы AS/400

9.1. Архитектура PowerPC - как основа системы AS/400

Архитектура PowerPC вполне обычна т. е. обладает всеми традиционными для RISC характеристиками; командами фиксированной длины, операциями «регистр- регистр», простыми режимами адресации и большим набором регистров. В то же время она имеет и характерные отличия.

Модель архитектуры PowerPC

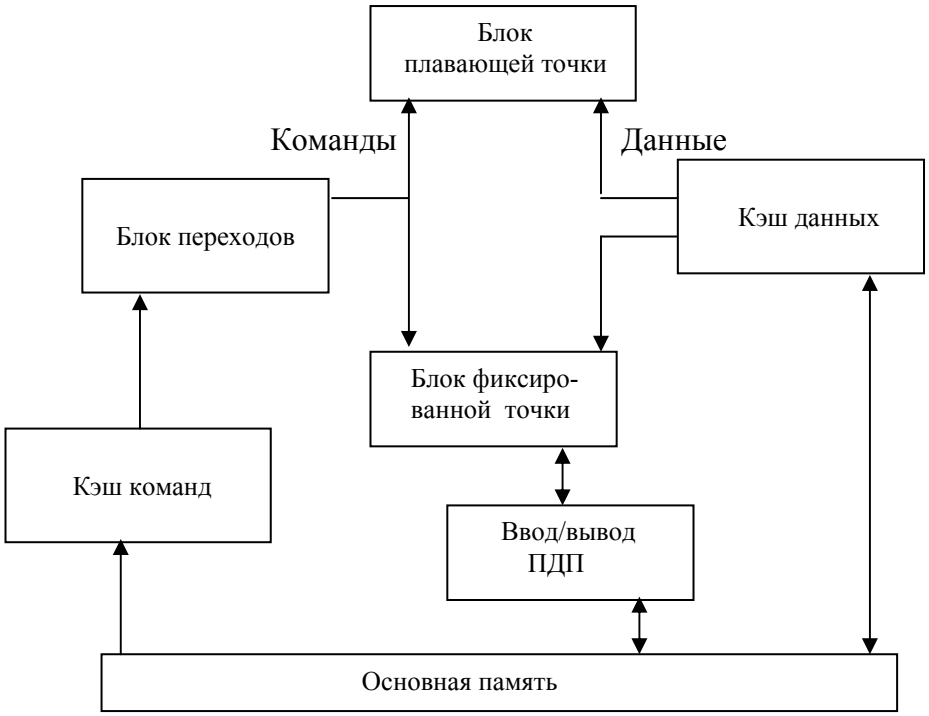


Рис. 9.1.

В основе всего набора команд лежит идея суперскалярной реализации. В суперскалярном процессоре за один такт несколько команд могут быть распределены на несколько конвейеров. Аппаратура процессора просматривает поток команд и отправляет на выполнение максимально возможное число независимых команд, обычно две — четыре за цикл. В дальнейшем эти команды могут выполняться параллельно и даже завершиться в порядке отличном от порядка их следования. Этот дополнительный параллелизм заметно повышает производительность процессора.

Команды направляются одновременно в три независимых исполняющих блока. Общая структура PowerPC представлена на рис. 9.1. Здесь показаны блоки переходов, фиксированной точки, плавающей точки, а также кэш команд, кэш данных, память и пространство ввода/вывода, которое в данной архитектуре выглядит как часть памяти.

Для каждого исполняющего блока архитектурой определен независимый набор регистров. Любая определенная архитектурой команда может выполняться только одним типом управляющих блоков. Таким образом, у каждого блока есть собственный набор регистров плюс собственный набор команд. Эти исполняющие блоки часто называют процессорами, так как им присущи все характеристики процессора. Можно сказать, что процессор PowerPC содержит три отдельных процессора - исполняющих блока. Заметьте также, что у каждого исполняющего блока может быть несколько несколько конвейеров команд. Если, например, модели для контролера сетевого интерфейса важна производительность операций с плавающей точкой, то блок плавающей точки должен содержать два конвейера и более и выполнять более одной команды плавающею точки одновременно. То же верно и для двух других блоков. Возможно создание процессоров PowerPC, способных сразу выполнять пять или более команд.

Преимущество такой схемы не только в возможности одновременного выполнения нескольких команд, но и в том, что, благодаря наличию у каждого блока отдельных ресурсов достаточно минимального объема взаимодействия и синхронизации между блоками. Исполняющие блоки способны подстраиваться под поток команд и позволять командам обгонять другу друга и завершаться в ином порядке.

Архитектура PowerPC отличается от обычного RISC-процессора еще и использованием нескольких составных команд.

Самый большой недостаток RISC в сравнении с CISC - объем кода. Для выполнения одной и той же программы RISC требуется больше команд, чем CISC. Составные команды позволяют минимизировать это разрастание кода. Некоторые из них весьма просты — например, обновление регистра базы при загрузке и сохранении, позволяющее исключить дополнительную команду прибавления. Другие, такие как команды множественной загрузки и сохранения, предназначенные для перемещения значений нескольких регистров одной командой, сложнее. Есть и команды загрузки/сохранения

цепочек, позволяющие загружать и сохранять произвольно выровненную цепочку байтов. В последней паре команд поклонники CISC различают не очень хорошие замаскированные команды пересылки символов.

Некоторые операции, вроде пересылки невыровненных строк байтов, происходят довольно часто и требуют определенной оптимизации. Если составная команда дает то, что нужно, но нарушает какое-то неписаное правило чистого RISC. Составные команды не знаменуют возврат к CISC-архитектуре — они лишний раз доказывают, что нет ничего абсолютно белого или черного.

Интенсивное применение суперскалярных возможностей и составных команд — основа философии проектирования архитектуры PowerPC. Эта философия используется и другими архитектурами, такими как Sun SuperSPARC: и Motorola 88110.

Что такое мегагерц? В последние годы стало популярно выражать производительность микросхемы процессора ЭВМ в мегагерцах. Эта единица характеризует тактовую частоту. Для простоты ее можно соотнести со скоростью вращения автомобильного двигателя: эта величина показывает, сколько оборотов в минуту совершает коленчатый вал. Скорость процессора можно задать как число тактов в секунду. За один число команд, которые процессор может выполнить в секунду, физическая единица герц (Гц), получившая свое название в честь немецкого физика, равна одному циклу в секунду, а один мегагерц — это миллион циклов в секунду.

Примерами этой философии высокой тактовой частоты являются архитектуры Digital Alpha, HP PA-RISC и MIPS R4000. Для сравнения возьмем процессор PA-RISC. Старшие модели PA-RISC: обычно выполняют две команды за такт, менее мощные модели PowerPC — три команды, а старшие модели - четыре или более. Этот дополнительный параллелизм дает PowerPC выигрыш в производительности, хотя и за счет увеличения сложности, что может снизить тактовую частоту.

Системы Speed Daemons (высокая тактовая частота) и Brainics (сложность). Основной вопрос в том, что тактовая частота, измеряемая в мегагерцах, не всегда адекватно отражает соотношение производительности процессоров 150-МГц Brainiac может легко превзойти по производительности 300- МГц Speed Daemon. Все зависит от выполняемой программы и степени параллелизма команд, достигаемой компилятором.

9.2. Расширения архитектуры PowerPC

Архитектура PowerPC определяет требуемые и необязательные команды как для 32-, так и для 64-разрядного набора команд, и каждый процессор PowerPC реализует разные наборы необязательных команд. Самое значительное расширение в архитектуре для AS/400 — поддержка тегов памяти.

В System/38 была введена концепция одноуровневой памяти. Попросту говоря, вся память, в том числе дисковая, является единым большим адресным

пространством. Нам требовался эффективный механизм для защиты областей памяти от пользователей, не имеющих к ним прав доступа. В MI адресация выполнялась через 16-байтовые указатели. Указатель содержит некоторый адрес, который пользователь может изменить. Поскольку измененный адрес указывает на любую область памяти, необходимо предоставить способ предотвращения неавторизованных изменений адресов пользователями.

С каждым словом памяти System/38 связан специальный бит защиты памяти — бит тега (tag bit). В слове памяти System/38 — 32 разряда данных. Указатель MI занимает четыре таких слова. Всякий раз, когда операционная система сохраняет в четырех последовательных словах памяти указатель, аппаратура включает (устанавливает в 1) 4 бита тега для индикации того, что указатель содержит адрес, допустимый для этого пользователя. Если пользователь изменяет в памяти любую часть указателя, то аппаратура выключает (устанавливает в О) бит тега. Если хотя бы один из битов тега сброшен, то адрес в указателе неверен и его нельзя использовать для доступа к памяти.

В целях безопасности бит тега необходимо скрыть: он должен храниться в недоступной пользователю области памяти. Бит тега не может быть одним из битов данных внутри слова, поскольку такие биты пользователь может видеть и изменять. Он должен храниться отдельно. System/38 использует для каждого слова памяти биты кода коррекции ошибок. Часть памяти с этими битами невидима программам, работающим поверх MI. Разработчики добавили к битам кода коррекции ошибок еще один бит и использовать его как бит тега. Если какая-либо пользовательская программа изменяет слово памяти, процессор должен автоматически сбрасывать скрытый бит тега. Если данное слово является частью указателя, то последний становится неверным. Только микрокод, расположенный ниже MI, имеет команды для включения битов тега.

AS/400 также использует биты тега в памяти. Поскольку в архитектуре PowerPC биты тега не предусмотрены, было добавлено к ней режим активных тегов (tags-active mode). В этом режиме процессор знает о наличии битов тега и будет сбрасывать их всякий раз, когда пользователь изменяет слово в памяти. Все процессоры AS/400 работают в режиме активных тегов, а процессоры PowerPC используют режим неактивных тегов.

9.3. 65 - разрядный процессор

В AS/400 ширина слова памяти возросла до 64 разрядов данных. Каждая восьмерка байтов памяти AS/400 связывает бит тега, и указатель MI занимает два таких слова. Разработчики полагали, что хранить два теговых бита в регистрах новых RISC-процессоров, как и в памяти, в некоторой степени выгодно. Кроме того, необходимо было сократить размер указателей MI до 8 байт. Внутри 16-байт указателей было неиспользуемое пространство, которое нужно сжать.

Чтобы хранить тестированные указатели в регистрах, размер целочисленных регистров нужно было увеличить до 65 разрядов. Эта схема разрабатывалась около года. Но от нее пришлось отказаться и вернутся к тому, чтобы хранить теги только в памяти. На то были три основные причины. Во-первых, изменение размера указателя влияло на OS/400 и требовало внесения в ее код слишком многих модификаций. Во-вторых, такой подход ограничивал будущие расширения размера адреса 64 разрядами. В-третьих, и это было главным, процессоры в режиме активных тегов оказались бы несовместимы с набором команд PowerPC.

Будущие процессоры, реализующие режим неактивных тегов, где 65-й разряд игнорируется, были бы полностью совместимы с PowerPC. Поначалу не собирались реализовывать 32-разрядные команды в режиме активных тегов, полагая, что этот режим будет использоваться только операционной системой AS/400, которая имеет дело лишь с 64-разрядными командами. На предварительном этапе выполнять какое-либо 32-разрядное программное обеспечение в режиме активных тегов не планировалось.

Затем, когда было решено поддерживать совместимость с набором команд PowerPC, пришлось избавиться от 65-го разряда в процессоре, полагая, что в дальнейшем возможно некое слияние операционных систем IBM. Поскольку большая часть программного обеспечения будет написана для 32-разрядного процессора, то и процессоры даже в режиме активных тегов реализуют 32-разрядный набор команд. Все будущие процессоры Рочестера при включенном режиме неактивных тегов смогут исполнять любые приложения и операционные системы для PowerPC.

Хотя вернулись к 64-разрядным процессорам уже много лет назад, в IBM по-прежнему иногда говорят о 65-разрядных процессорах, которые никто никогда не создавал. Просто многим неизвестно, что здесь делает бит тега. Вероятно, если бы мы назвали его битом защиты указателя в памяти (pointer in memory protection), то меньше бы людей пребывало в заблуждении. Увы, тогда нам пришлось бы все свое время посвятить объяснениям того зачем нам нужен бит-«pimp» .

9.4. Команды PowerPC AS/400

Архитектура PowerPC определяет привилегированные операции и команды, используемые не приложениями, а только операционной системой. Режим активных тегов включает разширения, добавленные для AS/400.

Например, механизм трансляции адреса должен поддерживать и одноуровневую память с общим адресным пространством, и обычную память с отдельным адресным пространством для каждого процесса. С помощью режима активных тегов мы приказываем процессору обратиться к одноуровневой памяти. В режиме неактивных тегов процессор использует обычную трансляцию адреса PowerPC.

В состав других расширений для AS/400 входят команды для работы с десятичными числами, некоторые новые команды загрузки и сохранения, а также расширения внутреннего регистра состояния процессора для усовершенствования переходов.

Следующие цифры помогут обобщить и изменения в архитектуре AS/400 и связать их с перспективой развития архитектуры PowerPC:

32-разрядная архитектура PowerPC определяет 187 команд, причем 11 из них необязательные;

64-разрядная архитектура PowerPC определяет 228 команд (187 из 32-разрядного набора плюс 41 дополнительная), из них 21 необязательная;

архитектура Amazon определяет 253 команды (228 из 64-разрядного набора PowerPC плюс 25 дополнительных), из них 20 необязательных. Причем 25 дополнительных команд доступны только в режиме активных тегов. Режим неактивных тегов поддерживает лишь 64-разрядный набор команд PowerPC.

Кроме того, определение любой архитектуры динамично и конкретные числа могут изменяться.

9.5.Процессоры AS/400

Два процессора, реализации которых здесь рассмотрим, поддерживают только режим активных тегов и только модель ввода/вывода AS/400. Это значит, что на них могут исполняться приложения, но не операционные системы, написанные для стандартного процессора PowerPC. Любая другая операционная система, исполняющаяся на одном из этих процессоров, для таких функций, как ввод/вывод, должна использовать средства, предоставляемые операционной системой AS/400.

В перспективе процессоры AS/400 будут поддерживать режимы как активных, так и неактивных тегов в дополнение к поддержке нескольких структур ввода/вывода одновременно, т. е. смогут исполнять любую операционную систему PowerPC.

9.5.1. Процессоры A3O (Muskie)

Разработанный в Рочестере под именем Muskie, A3O был выпущен в 1995 г. как старшая модель процессора для AS/400. В момент выхода A3O являлся самым быстрым из процессоров технологии PowerPC и самым быстрым микропроцессором IBM. Один такт этого высокопроизводительного процессора длится всего 6,5 нс., что соответствует тактовой частоте 754 МГц. A3O явно ориентирован на применение в системах для коммерческих расчетов, а не на технических рабочих станциях.

A30 - одномодульный многокристальный конвейерный процессор, предназначенный для старших моделей AS/400. Он способен выбирать и исполнять до четырех команд за такт. Пиковая производительность операций с фиксированной точкой (целочисленных) для A3O равна 616 млн. команд в секунду (MIPS - million instructions per second). Кроме того, процессор имеет

блок плавающей точки с пиковой производительностью 308 млн. команд в секунду (MFLOPS — million floating-point operations per second), 8-Кбайт кэш команд на кристалле, 256-Кбайт кэш модуля и поддерживает до 64 Гбайт основною памяти. Процессор поддерживает и многопроцессорные конфигурации.

Процессор А30 со всеми своими вспомогательными схемами занимает семь кристаллов, упакованных в один многокристальный модуль, насчитывающий более 25 млн. транзисторов. Один кристалл управляет вводом/выводом, т. е. технически не является частью процессора. Остальные шесть, составляющие процессор, с соединениями между ними показаны на рис. 9.2.

Структурная схема процессора А30

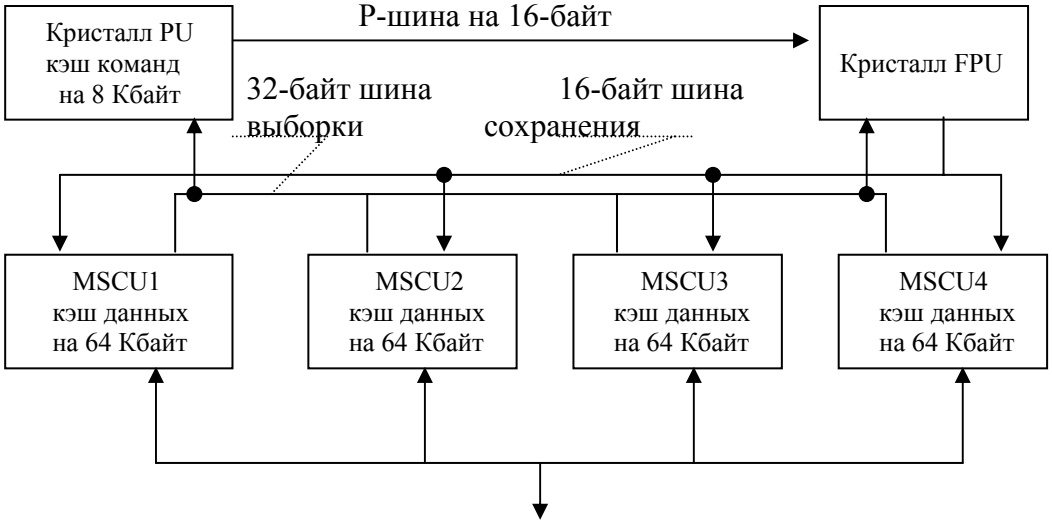


Рис. 9.2.

Однокристальные процессоры обычно изготавливаются по КМОП-технологии (комплементарная структура металл - оксид— полупроводник). Микросхемы КМОП потребляют меньше мощности (т. е. рассеивают меньше тепла), чем микросхемы других технологий. В результате на одном кристалле может быть больше транзисторов. Пока все схемы размещены на одном кристалле, маломощные цепи КМОП работают очень быстро. Другое дело - связи между кристаллами. Если для этого используются усилители КМОП, производительность падает.

Все шесть кристаллов А3О основаны на технологии БиКМОП (BICMOS, биполярный-КМОП). Биполярная технология, обеспечивая высокое быстродействие, требует большей мощности. Из-за объема рассеиваемого тепла биполярные кристаллы не могут использовать такую же высокую плотность упаковки, как в кристаллах КМОП. Преимущество технологии БиКМОП в

способности сохранять высокое быстродействие и при передаче между кристаллами. Это возможно за счет размещения на одном кристалле как биполярных цепей, так и цепей КМОП. Последние используются для схем внутри кристалла, а первые — для внешних усилителей.

БиКМОП отлично подходит для многокристального процессора, хотя и сопровождается большим выделением тепла. Для иллюстрации: семикристальный модуль АЗО рассеивает приблизительно 130 Вт. Можете сравнить это со 130-Вт электролампочкой, находящейся в коробке, длина сторон которой всего 63,5 мм. Без воздушного охлаждения такому модулю не обойтись - вот почему этот процессор используется только в модулях старших моделей AS/400 - 530 и 535.

Чтобы понять, почему для реализации процессора необходимо так много транзисторов, рассмотрим подробнее его кристаллы. В состав шести кристаллов одного процессора входят кристалл процессорного блока (PU — Processing Unit), Кристалл блока плавающей точки (FPU — Floating-Point Unit) и четыре одинаковых кристалла блоков управления основной памятью (MSCU — Main Store Control Unit).

На Кристалле PU расположены кэш команд, блок переходов и блок фиксированной точки. Кэш команд (8 Кбайт, ширина 32 байт) может выбирать из памяти за один такт 32 байт (восемь команд). Для передачи больших объемов данных за один такт предназначены 32-байт тракты данных. Даже регистровый стек блока фиксированной точки рассчитан на загрузку или сохранение четырех 64-разрядных регистров за такт.

FPU размещается на отдельном кристалле и поддерживает стандарт IEEE для операций с плавающей точкой. Этот блок способен выдавать результат в каждом такте, обеспечивая очень высокую производительность команд с плавающей точкой для данного процессора. Четыре команды за такт передаются от кристалла PU на кристалл FPU по 16-байт P-шине. Вся информация, хранящаяся в памяти, также пересылается из PU по P-шине в кэш данных, откуда она выбираются для FPU со скоростью 32 байт/такт. Данные из FPU и PU пересылаются в кэш данных по 16-байт шине записи.

MSCU реализует кэш данных и интерфейс памяти. Все четыре кристалла совместно предоставляют 256-Кбайт кэш и интерфейсы к шинам данных (см. рис. 8.2.). При конвейерных обращениях к кэшу за один такт считывается 32 и сохраняется 16 байт данных. MSCU поддерживает многопроцессорные конфигурации, обеспечивая когерентность кэшей разных процессоров.

PU и FPU вместе насчитывают пять конвейеров, однако в каждом цикле может быть распределено только четыре команды:

перехода (включая операцию над содержимым регистре условия);

загрузки/сохранения;

арифметики с фиксированной точкой;

фиксированной точки для логических операций сдвига или циклического сдвига, или команда плавающей точки.

Команды плавающей точки исполняются в FPU, но не одновременно с выполняемыми в PU командами фиксированной точки для логических операций, сдвига или циклического сдвига. Конвейер загрузки/сохранения осуществляет выборку и запись данных как с фиксированной, так и с плавающей точкой. Наличие не скольких конвейеров позволяет выполнять фрагменты нескольких команд одновременно.

Вся описанная выше аппаратура составляет полностью 64 разрядный процессор, работающий с тактовыми частотами 125 и 154 МГц. Процессор АЗО также поддерживает общую память и симметричное мультипроцессирование (SMP). Первоначально поддерживаются конфигурации до четырех процессороз, но возможны и более крупные.

Являясь самым быстрым для своего времени RISC-процессором IBM АЗО оптимизирован для нужд коммерческих вычислений. Иллюстрируют это положение следующие характеристики.

Комерческие системы и серверы должны обрабатывать огромные объемы информации. 16-байт (128-бит) и 32-байт (255 бит) шины позволяют этому процессору справляться с большими объемами данных и команд. Сравните это с типичными 8-байт (64-бит) шинами большинства высокопроизводительных RISC процессоров. Последние предназначены для использования в рабочих станциях, где обрабатываются гораздо меньшие объемы данных.

Даже в системе способной перемещать большие объемы данных кэш обычно является узким местом большинства RISC-процессоров. Для его ликвидации в АЗО предусмотрен 256-Кбайт кэш данных, работающий за один цикл. Производительность кэша здесь доходит до 4,9 Гбайт/с, а производительность шины - до 2,2 Гбайт/с. Это вдвое превосходит быстродействие других высокопроизводительных RISC-процессоров, предназначенных для технических расчетов.

Поскольку команда перехода может вызвать простои конвейера, современные RISC-процессоры подобно суперЭВМ, реализуют некоторую разновидность предсказания переходов. Для большинства RISC-процессоров точность предсказания переходов при выполнении технических задач составляет 80 - 90%. Эти высокие показатели достигаются благодаря тому, что в технических задачах большой объем циклической обработки, когда процессор несколько раз повторяет последовательность команд тела цикла. Для программ подобного типа функция предсказания переходов работает отлично. В программах для коммерческих задач циклов гораздо меньше и точность предсказания переходов здесь может быть ниже 50% (это соответствует точности случайного выбора). Поэтому, вместо того чтобы пытаться угадать место перехода, АЗО выбирает команды из обоих мест перехода и начинает выполнять их (так называемое спекулятивное выполнение — speculative execution). Данный метод при наличии очень скоростного кэша (а

он у АЗО есть) позволяет достичь, по сути, 100%-ной точности на задачах любого типа.

Еще один важный аспект коммерческих вычислений — высокая степень целостности данных и высокий коэффициент готовности. АЗО реализует коды коррекции ошибок для всех связей за пределами кристаллов. Кроме того, большая часть логики управления и передачи данных на каждом кристалле также содержит различные схемы контроля. Сравните это с типичным RISC-процессором для рабочей станции, который редко имеет что-либо подобное этим возможностям определения и исправления ошибок.

9.5.2. Процессоры A10 (Cobra)

В процессорах A10, как и в АЗО, реализована расширенная 64-разрядная архитектура PowerPC. Кроме того, A10 являются суперскалярными, что позволяет использовать параллелизм на уровне команд. Функционально оба семейства процессоров исполняют один и тот же набор команд уровня приложений. Реализуемые ими необязательные команды несколько различны. Например, A10 предназначается для средних и младших моделей AS/400, поэтому команды, поддерживающие такие возможности, как мультипроцессирование, в него не включены.

Процессоры A10 разработаны в Эндикотте, штат Нью-Йорк. В настоящий момент имеется четыре модели процессоров Cobra. В системах RISC AS/400 используются Cobra-4 и Cobra-CR (CR — cost reduced, т. е. цена снижена). Cobra-CR - это процессор Cobra-4, способный работать только на частоте 50 МГц - самой низкой частоте процессора Cobra-4.

Для тестирования нового программного обеспечения операционной системы команда проектировщиков в Эндикотте предложила специальный вариант под названием Cobra-0. Он используется только для тестирования и не устанавливается в AS/400 Небольшая группа из Рочестера разработала четвертую версию для системы Advanced 36, анонсированной в 1994 г. Она названа Cobra-Lite, поскольку в ней отсутствуют 17 обязательных команд PowerPC.

При разработке A10 ставилась задача объединить процессор и интерфейс памяти на одном кристалле (интерфейс шины ввода/вывода располагается на отдельном кристалле). Для достижения этого A10 использует технологию КМОП, а не БиКМОП. Точнее, в A10 применяется технология, названная IBM CMOS-45. В результате процессор рассеивает меньше тепла, чем АЗО, и может устанавливаться в корпусах меньшего размера с меньшими требованиями к охлаждению. Поэтому оригинальные корпуса, представленные для Advanced Series в 1994 г., оснащаются только процессорами A10.

Подобно АЗО, процессор A10 имеет пять конвейеров, но за один цикл может распределять не более трех команд:

перехода (включая команду регистра условия);

загрузки/сохранения;

арифметики с фиксированной точкой (включая логические команды, команды сдвига и циклического сдвига); или команду плавающей точки; или команду регистра условия.

Три конвейера (фиксированной точки, плавающей точки и команд регистра условия) совместно используют третий слот распределения.

Первые процессоры A10 работают на тактовых частотах 50 и 77 МГц, но их конструкция допускает и более высокие частоты. На частоте 77 МГц A10 показывает 231 MIPS. Для поддержания подобной скорости A10 имеет 4-Кбайт внутренний (на кристалле) кэш команд и 8-Кбайт (также внутренний) кэш данных. Эти кэши могут быть дополнены внешним (на отдельных микросхемах) кэшем в 1 Мбайт.

9.6. Машинный интерфейс AS/400 (MI)

Сравнивая с обычным машинным интерфейсом, мы часто говорим о нем как о машинном интерфейсе высокого уровня. Дело в том, что многие инструкции выполняют очень сложные функции, в то время как мало какие из существующих машинных интерфейсов располагают инструкциями вызова, поддерживающими и раннюю и позднюю компоновку. Для обычного интерфейса более характерна наличие инструкций передачи управления.

Чтобы понять разницу инструкцию обычного машинного интерфейса (рис. 9.3). Она состоит из кода операции (КОП) и одного или нескольких полей оперрандов. Инструкции могут быть арифметическими (в каждом компьютере есть инструкция сложения), передачи управления и манипуляции данными. Самое важное, с операндами какого рода имеют дело инструкции.

Обычные машинные интерфейсы работают с содержимым регистров, памяти или непосредственно с данными, записанными в самой инструкции. Иначе говоря, они не подозревают о существовании данных приложения или операционной системы. Возьмем такую стандартную инструкцию, как сложение содержимого регистров. Она определяет два регистра процессора и выполняет операцию, извлекая биты из одного регистра, складывая их с битами из другого регистра и размещая результат в определенном месте. Биты для инструкции значения не имеют. Об их значении заботится программа, но не инструкция. Машина не знает, что там лежит, - это просто набор битов, к которому применяется алгоритм сложения. То, что в регистрах находятся имена двух сотрудников и поэтому рассматривать их в качестве арифметических операндов нет смысла, не учитывается. Операции этого уровня представляет собой просто обработку содержимого регистров или памяти.

Обычный машинный интерфейс

Рис. 9.3.

Мы уже говорили о недостатке такой структуры - ее существенной зависимости от аппаратной технологии. Так как инструкции работают в адресном пространстве, пространстве регистров ввода/вывода и пространстве регистров, они привязаны к этим физическим структурам. Их изменение может потребовать изменения инструкций. Значит, преобразование существующих программ связано с большими проблемами.

Машинный интерфейс AS/400 (рис. 9.4) устроен совсем иначе. У него, как и у обычных машин, имеется набор инструкций с кодами операций и операндами. Есть в нем также разные типы арифметических операций (команды сложения) и операций передачи управления, работающих с традиционными операндами. Отличается же он от обычной машины инструкциями, аналогичными промежуточному представлению, характерному для современных компиляторов языка высокого уровня, а также структурами данных (объектами).

Главное здесь не в самих инструкциях или операциях, а в использовании ими операндах. В обычной машине есть регистры, память и непосредственные данные. В MI тоже имеются непосредственные данные, но нет ни регистров, ни памяти. Их заменяют объекты.

Рис. 9.4.

В машинном интерфейсе MI определены объекты нескольких типов. Большинство из них - сложные структуры данных, необходимые для представления информационных ресурсов. Одним из самых важных типов объектов в системе служит пространство – просто набор байтов не относящийся к физическому оборудованию. Многие с трудом представляют себе, что значит масса подвешенных неизвестно где байтов, которую очень хочется связать с аппаратурой. Но в MI понятие пространства не имеет отношения к физической памяти – оно абсолютно не зависимо от того, что находится ниже.

Когда программе MI требуется память, она использует пространство. Это не концепция регистров и физической памяти и не адресное пространство в традиционном смысле. Например, компилятор AS/400 помещает в пространство созданный шаблон программы.

Кроме пространства, существуют и другие типы объектов. До сих пор мы обсуждали только системные объекты MI. Но объекты поддерживают и OS/400.

10.Принципы организации вычислительных систем

10.1. Классификация вычислительных систем

Создание вычислительных систем (ВС)—наиболее реальный путь разрешения противоречия между непрерывно растущими потребностями в быстродействующих и надежных средствах вычислений и пределом технических возможностей ЭВМ на данном этапе развития.

Вычислительная система представляет собой сложный комплекс, состоящий из разнообразных технических средств соответствующего программного обеспечения. Как технические, так и программные средства имеют модульную структуру построения, позволяющую наращивать ее в зависимости от назначения и условий эксплуатации системы. Программная автоматизация управления вычислительным процессом осуществляется с помощью ОС.

Первыми ВС были однопроцессорные мультипрограммные ЭВМ, высокая производительность которых была достигнута за счет распределения во времени основных устройств системы между программами.

Дальнейшее повышение производительноти ЭВМ было достигнуто за счет мультиобработки программ (задач), т. е. за счет разбиения программ на отдельные блоки и параллельной обработки этих блоков на нескольких обрабатывающих устройствах, входящих в состав ВС. Мультиобработка позволяет не только повысть производительность, но и сократить время выполнения отдельных программ, которые могут разбиваться на части и распределяться между различными обрабатывающими устройствами.

Первым типом ВС с мультиобработкой был многомашинный комплекс МК - многомашинная ВС. В состав МК объединялись различные ЭВМ с классической структурой, имеющие возможность обмениваться информацией.

На рис. 10.1 представлена структура двухмашинной ВС. Каждая ЭВМ имеет ОП, ВЗУ, ПфУ, подключаемые к центральной части ЭВМ - процессору (ПР) с помощью каналов ввода-вывода (КВВ), и работает под управлением своей ОС. Обмен информацией между ЭВМ1 и ЭВМ2 осуществляется через системные средства обмена (ССО) в результате взаимодействия ОС машин между собой.

Основной недостаток многомашинной ВС - недостаточно эффективно используется оборудование комплекса. Достаточно, в ВС в каждой ЭВМ выйти из строя по одному устройству (даже разных типов), как вся ВС становится неработоспособной.

Структура многомашинной ВС

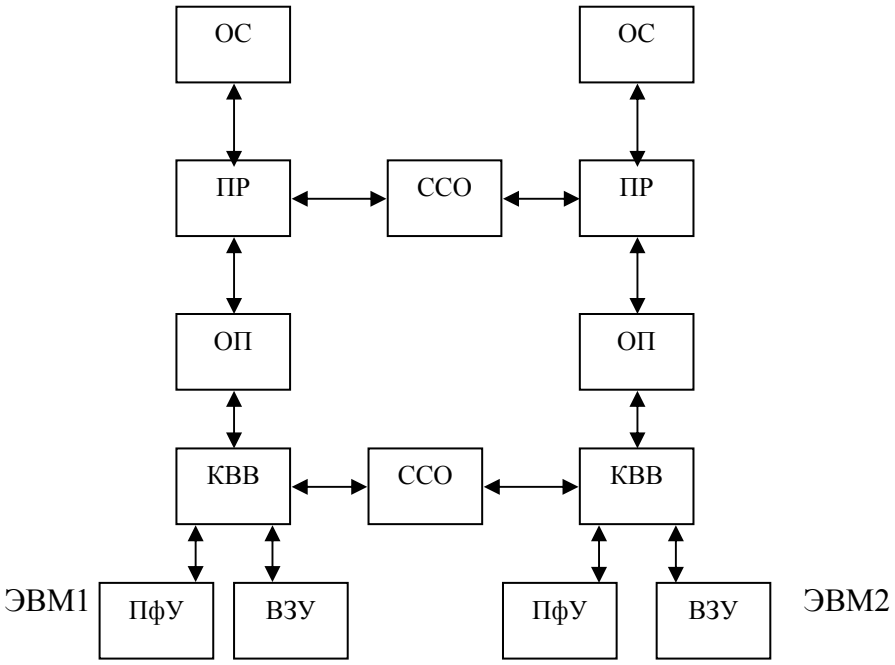


Рис.10.1.

Следующим шагом в направлении дальнейшего увеличения производительности ВС явилось создание многопроцессорных ВС с мультиобработкой, в составе которых содержится два или несколько процессоров (ПР), работающих с единой ОП, общий набор каналов ввода-вывода (КВВ) и ВЗУ (рис. 10.2). Наличие единой ОС делает возможным автоматическое распределение ресурсов системы на различных этапах ее работы. В результате достигается высокая «живучесть» ВС, позволяющая в случае отказа отдельных. модулей перераспределить нагрузку между работоспособными, обеспечив тем самым выполнение наиболее важных для ВС функций.

К недостаткам многопроцессорных ВС относят трудности, возникающие при реализации общего поля ОП, ВЗУ, а также при разработке специальной ОС.

Дальнейшее развитие идей мультиобработки привело к созданию крупных многопроцессорных систем высокой производительности, получивших назначение высокопараллельных ВС. Такие ВС в зависимости от ее структуры могут одновременно обрабатывать множественный поток данных или команд. Под потоком команд понимается последовательность команд, выполняемых ВС, а потоком данных - последовательность данных, обрабатываемых под управлением потока команд.

Структура многопроцессорной ВС

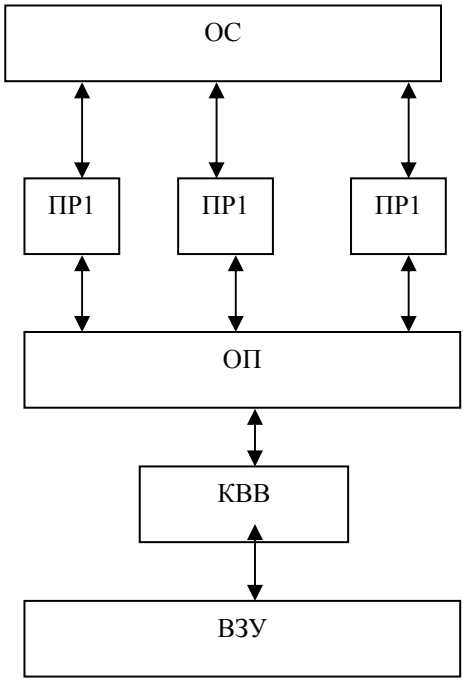


Рис. 10.2.

Высокопараллельные ВС структуры типа ОКМД (одиночный поток команд и множественный поток данных) получили название матричных ВС (рис. 10.3). Они содержат которое количество одинаковых сравнительно простых быстродействующих процессоров (ПР), соединенных друг с другом так, что образуется сетка (матрица), в узлах котрых размещаются ПР. Все НР выполняют одну и ту же команду, но над разными операндами, доставляемыми процессорам из памяти несколькими потоками данных.

Многопроцессорная ВС структуры типа ОКМД

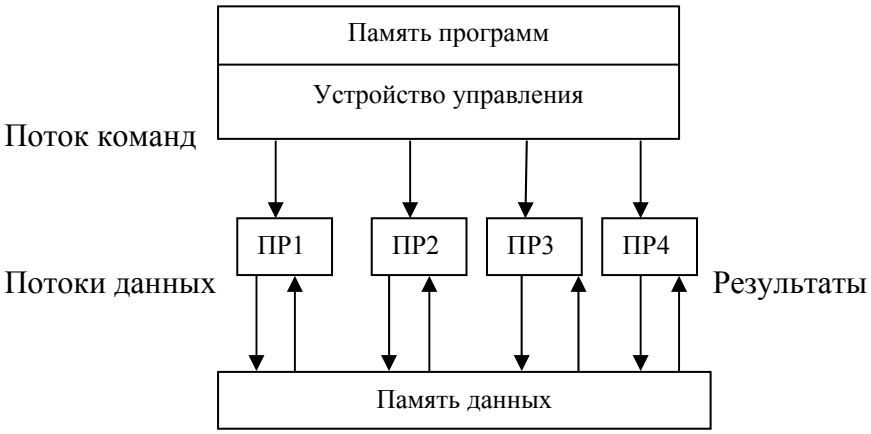


Рис. 10.3.

Высокопараллельные ВС структуры типа МКОД (множественный поток команд и одиночный поток данных) получили название конвейерных. ВС. Такие ВС (рис. 10.4) содержат цепочкку последовательно соединеных ПР, так что информация на выходе одного ПР является входной информацией для другого ПР. Каждый ПР обрабатывает соответствующую часть задачи, передавая результаты соседнему ПР, который использует их в качестве исходных данных.

Многопроцессорная ВС структуры типа МКОД

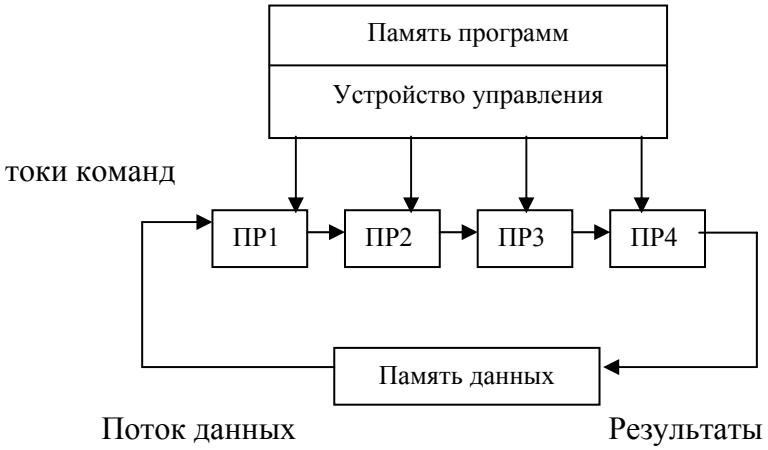


Рис. 10.4.

Так, например, операция сложения чисел с плавающей запятой может быть разделена на 4 этапа: сравнение порядков; выравнивание порядков; сложение мантисс; нормализация результата. В конвейерной ВС все эти этапы вычислений будут выполняться отдельными процессорами, образующими конвейер.

Высокопараллельные ВС по сравнению с многопроцессорными ВС обеспечивают более высокую производительнсть, надежность, и "живучесть". Однако при этом усиливаются недостатки - усложение управление системой, трудность программирования и малая загрузка системы.

Первые два недостатка компенсируются благодаря применению БИС и специальных языков программирования. Третий недостаток приводит к тому, что большинство высокопараллельных ВС ориентируется на специализированное применение.

Реализация идей мультипрограммной работы потребовала пересмотра способов взаимодействия оператора с программой. Прежде всего возникла необходимость в большом количестве абонентских пунктов, позволяющих нескольким пользователям одновременно взаимодействовать с ЭМВ по своим программам. Часто такие пульты находятся на значительном расстоянии от ЭВМ и связаны с ней телефонными или телеграфными каналами. В качестве оконечных устройств (терминалов) обычно используют дисплеи. Такая связь

ЭВМ со многими дистанционно расположенными пультами получила название мпогопультовой.

Многопультовый режим работы, обеспечивающий одновременный доступ к ЭВМ многих пользователей, вызвал соответствующую организацию их обслуживания. Обычно такое обслуживание осуществляется в режиме разделения времени между пользователями.

Вычислительная система с разделением времени последовательно и циклически опрашивает терминалы всех пользователей, записывает информацию обратившихся абонентов и обслуживает их в той же последовательности. Каждому пользователю (абоненту) выделяется определенный, строго ограниченный квант машинного времени (максимум доли секунд). В течение таких циклически повторяющихся квантов времени осуществляется выполнение каждой программы. В идеальном случае интервал времени между включениями в работу одной и той же программы не должен превышать обычное время реакции человека. Тогда программист, управляющий своей программой, не будет ощущать прерывистого характера ее выполнения и у него создается иллюзия индивидуального общения с ВС.

Вычислительные системы с разделением времени являются основой сети вычислительных центров коллективного пользования.

Чтобы удовлетворять предъявляемым требованиям к ВС, они должны иметь:

развитую ОС, обеспечивающую одновременное выполнение различных программ и организующую доступ пользователя к стандартным программам;

трансляторы с языков программирования, облегчающих работу программистов по подготовке программ;

средства, обеспечивающие динамическое распределение памяти между программами, а также свободное перемещение программ в процессе вычислений;

средства защиты памяти и программ от вмешательства других программ;

датчик времени (таймер), позволяющий в соответствии с запросами пользователей выделять им необходимое время для работы, по истечении которого ВС автоматически переключается на выполнение других программ;

как аппаратурные, так и программные средства с целью организации приоритетов для одновременно ждущих программ.

Многопроцессорные и многомашинные ВС классифицируют по различным признакам. Рассмотрим некоторые из них.

По назначению ВС делятся на универсальные и специализированные. Универсальные ВС предназначены для решения широкого круга задач, специализированные — для решения определенного круга задач. Специализированные ВС, как правило, должны иметь аппаратурные и программные средства, предназначенные специально для этой системы.

По типу оборудования ВС подразделяются на однородные и неоднородные. Однородные системы содержат несколько однотипных ЭВМ

(или процессоров), неоднородные - разнотипные ЭВМ (или процессоры). Основной недостаток однородных ВС - неполная загруженность отдельных ЭВМ (процессоров) во время ее работы. В целях повышения эффективности использования ЭВМ (процессоров) используются неоднородные ВС. Например, более производительная центральная ЭВМ системы выполняет обработку информации, а менее производительные ЭВМ1, ЭВМ2 и ЭВМ3 осуществляют ввод и вывод ннформации, ее предварительную. обработку и передачу в центральную ЭВМ. Коммутатор при передаче информации в центральную ЭВМ и выдаче из нее результатов настраивается на выбор соответствующей ЭВМ 2-го уровня (рис. 10.5).

Вычислительные системы с иерархической структурой могут иметь и более двух уровней иерархии. ЭВМ, выполняющие предварительную обработку информации, часто, называют машинами-сателлитами.

По типу структуры ВС разделяются с постоянной и переменной структурами. Под структурой ВС понимают состав системы и схемы функциональных и управляющих связей между ее элементами. В системах с постоянной структурой в процессе ее функционирования не изменяется состав функциональных и управляющих связей между ее элементами. Переменную структуру имеют адаптивные системы, т. е. такие, у которых структура изменяется на основе анализа текущей информации. Подобные системы позволяют достичь оптимального состояния в любых изменяющихся условиях функционирования.

По степени централизации управления ВС разделяются на централизованные, децентрализованные и со смешанным управлением.

Иерархическая структура многомашинной ВС

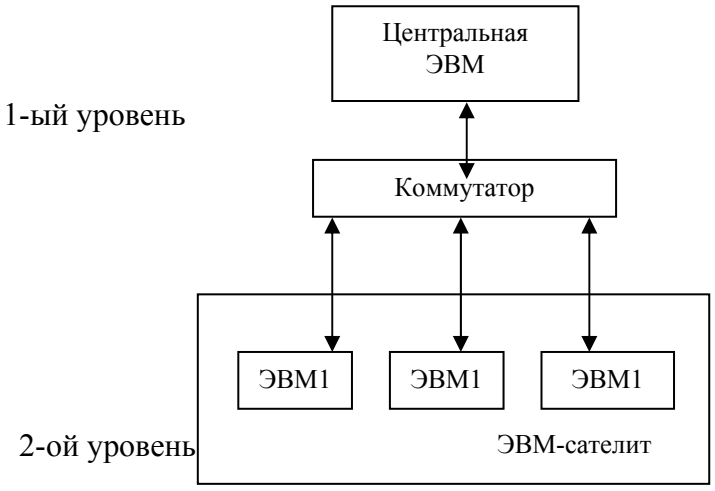


Рис. 10.5.

В централизованных ВС все функции управления сосредоточены в одном элементе, в качестве которого используется одна из ЭВМ, называемая машиной-директором, или центральный процессор.

В децентрализованной ВС каждый процессор или ЭВМ действуют автономно, решая свои задачи.

В системе со смешанным управлением ВС разбивается на группы взаимодействующих ЭВМ (или процессоров), в каждой из которых осуществляется централизированное управление, а между группами - децентрализированное.

10.2. Многомашинные вычислительные системы

В настоящее время наиболее широко используют двухмашинные вычислительные комплексы, которые могут работать в одном из следующих режимов.

1. 100% - ное горячее резервирование. Обе ЭМВ в этом режиме исправны и работают параллельно,выполняя одни те же операпни над одной и той же информацей (дуплексный режим). После выполнения каждой команды результаты преобразования сравниваются и при их совпадении процесс вычислений продолжается. При этом в памяти обоих ЭВМ в каждый момент находится одна и та же информация. При обнаружении несовпадения в результатах обработки Неисправная ЭВМ выводится на ремонт, а исправная ЭВМ продолжает работать под контролем встроенной в ЭВМ системы автоматического контроля.

2. 0дна исправная ЭВМ решает задачи без дублирования, а другая ЭВМ находится в режиме «Профилактика», в котором осуществляется прогон контролирующих тестов. Если основная ЭВМ продолжает не в состоянии выполнить задачу, то резервная может прекратить "Профилактику" и начать работу параллельно с основной.

3. Обе ЭВМ работают в автономном режиме со своим набором ПфУ по автономным рабочим программам.

Задание режимов работы вычислительного комплекса возможно программным путем или с помощью команд прямого управления или с пульта управления комплекса.

По типу организации многомашинные ВК можно разделить на две группы: незвязанные и связанные.

Несвязанные ВК разрабатывались с целью разгрузить центральный процессор от выполнения операций по вводу-выводу данных извне. Они состоят из центральной и периферийной ЭВМ, между которыми нет прямого физического соединения и отсутствуют какие-либо совместно исполняемые аппаратные средства. Целесообразность их применения определяется тем, что операции ввода-вывода информации и вычисления совмещаются во времени. Небольшая и недорогая ЭВМ выполняет медленные операции ввода-вывода

информации (считывание с перфокарт, печать и т. и.), а центральная ЭВМ — высокоскоростные операции, обмениваясь в процессе вычислений с ВЗУ.

Связанные ВК включают несколько ЭВМ, которые совместно используют общие аппаратные средства, т. е. в этих ВС возможно электрическое сопряжение между процессорами. В таких ВК, обе ЭВМ могут выполнять, две различные программы автономно или во взаимодействии друг с другом.

Наличие нескольких тесно связанных ЭВМ в составе единой ВС позволяет существенно уменьшить время вычислений благодаря параллельному выполнению на отдельных ЭВМ различных подзадач (пакетов программ), входящих в общую задачу. 0сновное условие эффективного использавания таких ВК- координация работы всех ЭВМ с помощью управляющей программы ОС, которая составляет список подзадач, подлежащих решению, и распределяет их между ЭВМ. Благодаря возможности передачи не только числовой, но и командной информации между ЭВМ можно передавать программы и части программ. Обмен програмами значительно упрощает управление ВК и позволяет при загрузке какой-либо ЭВМ часть нагрузки передать другой ЭВМ, упрощает создание библиотеки стандартных программ, пригодных для любой машины ВК.

Для реализации межмашинной связи могут испозоваться как средства, имеющиеся в составе ЭВМ, так и средства, предусмотренные специально для работы в составе данного ВК.

10.3. Многопроцессорные вычислительные системы

В настоящее время особое внимание уделяется созданию многопроцессорных ВК. Основной целью при разработке таких ВК является повышение производительности систем за счет: обеслечения возможности параллельного выполния независимых задач; повышения эффективности работы и улучшения распределения нагрузки в системе; обеспечения наиболее экономичного обслуживания экстренных заданий и заданий при пиковых нагрузках; достижения высокого коэффициента эффективного использования ресурсов для создания новых типов архитектуры комплекса.

В многопроцессорных ВС при решении задач с небольшими емкостями памяти возможно одновременное решение на разных процессорах. Если в какой-либо интервал времени требуется резкое увеличение емкости памяти, то вся память отдается для решения задачи.

Основные особенности построения многопроцессорных ВК заключаются в следующем:

система включает в себя один или несколько процессоров;

центральная память системы должна находиться в общем пользовании и к ней должен быть обеспечен доступ от всех процессоров системы;

система должна иметь общий доступ ко всем устройствам ввода-вывода, включая каналы;

система должна иметь единую ОС, управляющую всеми аппаратными и программными средствами;

в системе должно быть предусмотрено взаимодействие элементов аппаратного и программного обеспечения на всех уровнях: на уровне системного программного обеспечения, на программном уровне при решении задач пользователей (возможность перераспределения заданий), на уровне обмена данными и др.

В многомашинных ВС связь может осуществляться только на информационном уровне.

Важнейшее значение для организации многопроцессорной ВС имеют способы соединения между собой различных функциональных блоков системы, так как эффективность такой системы определяется степенью параллельности или совмещения по времени работы всех устройств системы.

11. Организация сетей

ЛВС могут состоять из одного файл-сервера, поддерживающего небольшое число рабочих станций, или из многих файл-серверов и коммуникационных серверов, соединенных с сотнями рабочих станций. Некоторые сети спроектированы для оказания сравнительно простых услуг, таких, как совместное пользование прикладной программой и файлом и обеспечение доступа к единственному принтеру. Другие сети обеспечивают связь с большими и мини-ЭВМ, модемами коллективного пользования, разнообразными устройствами ввода/вывода (графопостроителями, принтерами и т. д.) и устройствам памяти большой емкости (диски типа WORM).

11.1. Файл–сервер и рабочие станции

Файл-сервер является ядром локальной сети. Этот компьютер (обычно высокопроизводительный мини-компьютер) запускает операционную систему и управляет потоком данных, передаваемых по сети. Отдельные рабочие станции и любые совместно используемые периферийные устройства, такие, как принтеры, - все подсоединяются к файл-серверу.

Каждая рабочая станция представляет собой обычный персональный компьютер, работающий под управлением собственной дисковой операционной системы (такой, как DOS или OS/2). Однако в отличие от автономного персонального компьютера рабочая станция содержит плату сетевого интерфейса и физически соединена кабелями с файлом-сервером. Кроме того, рабочая станция запускает специальную программу,

называемой оболочкой сети, которая позволяет ей обмениваться информацией с файл-сервером, другими рабочими станциями и прочими устройствами сети. Оболочка позволяет рабочей станции использовать файлы и программы, хранящиеся на файл-сервере, так же легко, как и находящиеся на ее собственных дисках.

11.2. Операционная система рабочей станции

Каждый компьютер рабочей станции работает под управлением своей собственной операционной системы (такой, как DOS или OS/2). Чтобы включить каждую рабочую станцию с состав сети, оболочка сетевой операционной системы загружается в начало операционной системы компьютера.

Оболочка сохраняет большую часть команд и функций операционной системы, позволяя рабочей станции в процессе работы выглядеть как обычно. Оболочка просто добавляет локальной операционной системе больше функций и придает ей гибкость.

11.3. Топология локальных сетей

Термин "топология сети" относится к пути, по которому данные перемещаются по сети. Существуют три основных вида топологий: "общая шина", "звезда" и "кольцо".

Топология "общая шина" предполагает использование одного кабеля, к которому подключаются все компьютеры сети (рис. 11.1). В случае "общая шина" кабель используется совместно всеми станциями по очереди. Принимаются специальные меры для того, чтобы при работе с общим кабелем компьютеры не мешали друг другу передавать и принимать данные.

В топологии "общая шина" все сообщения, посылаемые отдельными компьютерами, подключенными к сети. Надежность здесь выше, так как выход из строя отдельных компьютеров не нарушит работоспособности сети в целом. Поиск неисправностей в кабеле затруднен. Кроме того, так как используется только один кабель, в случае обрыва нарушается работа всей сети.

На рис. 11.2 показаны компьютеры, соединенные звездой. В этом случае каждый компьютер через специальный сетевой адаптер подключается отдельным кабелем к объединяющему устройству.

При необходимости можно объединять вместе несколько сетей с топологией "звезда", при этом получаются разветвленные конфигурации сети.

С точки зрения надежности эта топология не является наилучшим решением, так как выход из строя центрального узла приведет к остановке всей сети. Однако при использовании топологии "звезда" легче найти неисправность в кабельной сети.

Используется также топология "кольцо" (рис. 11.3). В этом случае данные передаются от одного компьютера к другому как бы по эстафете. Если

компьютер получит данные, предназначенные для другого компьютера, он передает их дальше по кольцу. Если данные предназначены для получившего их компьютера, они дальше не передаются.

Локальная сеть может использовать одну из перечисленных топологий. Это зависит от количества объединяемых компьютеров, их взаимного расположения и других условий. Можно также объединить несколько локальных сетей, выполненных с использованием разных топологий, в единую локальную сеть. Может, например, древовидная топология.

Шинная (линейная) топология

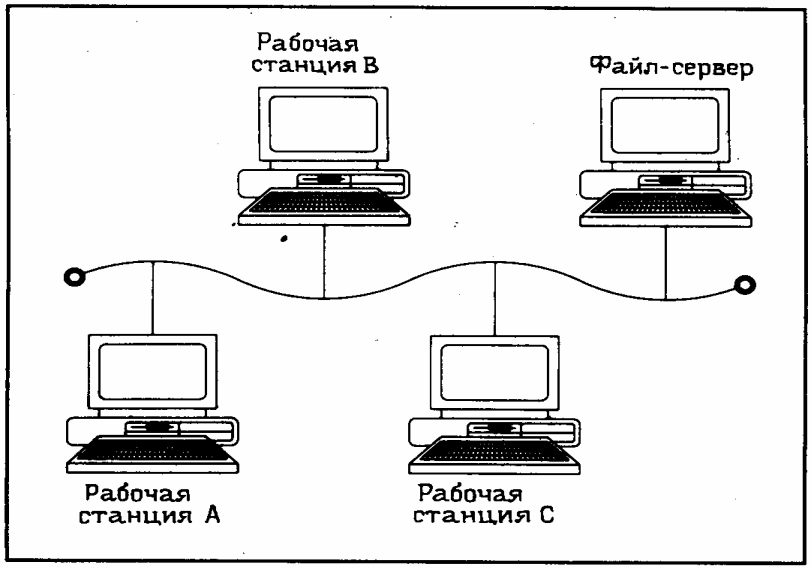


Рис. 11.1.

Топология типа «Звезда»

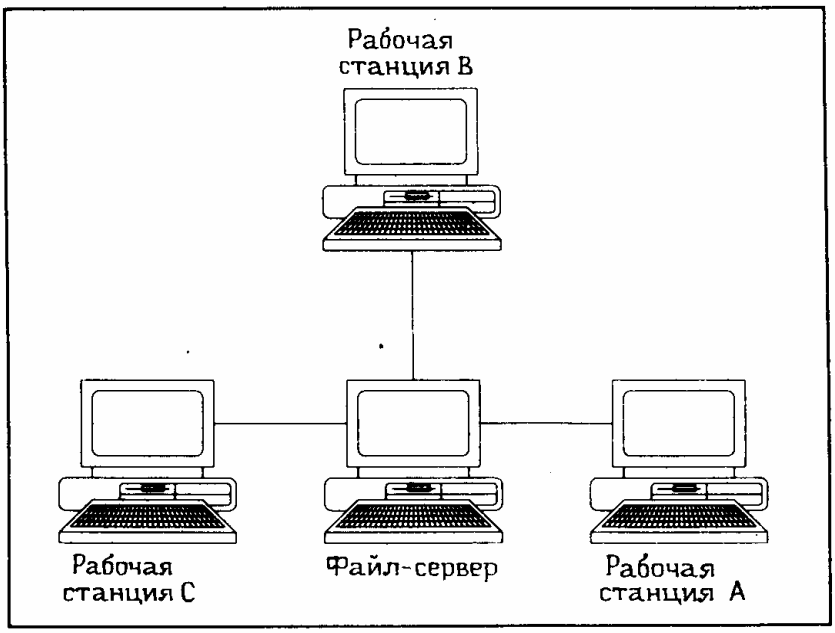


Рис. 11.2.

Кольцевая топология

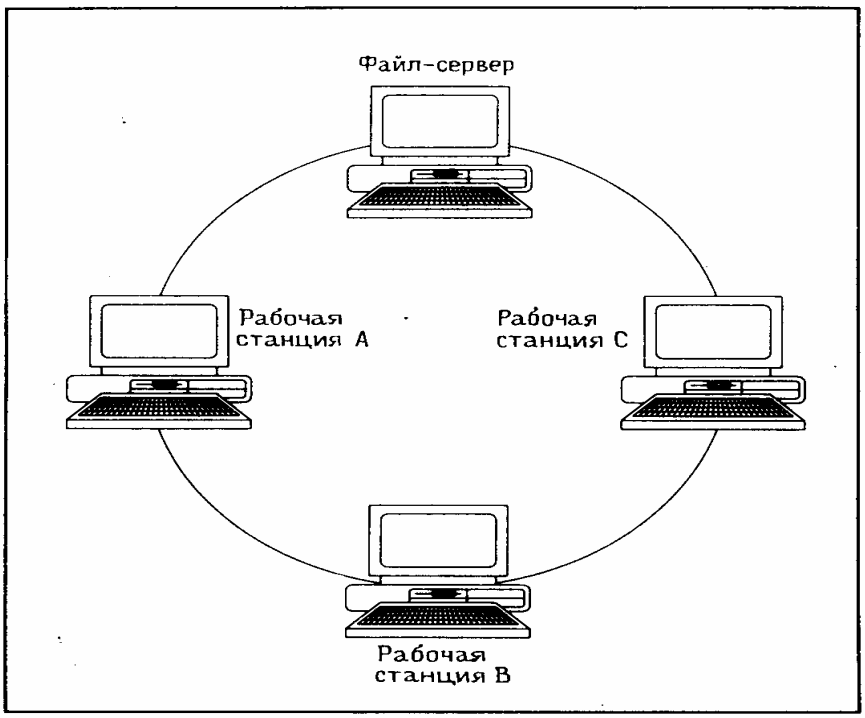


Рис. 11.3.

11.4. Методы доступа и протоколы передачи данных

В различных сетях существуют различные процедуры обмена данными в сети. Эти процедуры называются протоколами передачи данных, которые описывают методы доступа к сетевым каналам данных.

Наибольшее распространение получили конкретные реализации методов доступа: Ethernet, Arcnet и Token-Ring.

11.4.1. Метод доступа Ethernet

Это метод доступа, разработанный фирмой Xerox в 1975 году, пользуется наибольшей популярностью. Он обеспечивает высокую скорость передачи данных и надежность.

Для данного метода доступа используется топология "общая шина". Поэтому сообщение, отправляемое одной рабочей станцией, принимается одновременно всеми остальными, подключенными к общей шине. Но сообщение, предназначенное только для одной станции (оно включает в себя адрес станции назначения и адрес станции отправителя). Та станция, которой предназначено сообщение, принимает его, остальные игнорируют.

Метод доступа Ethernet является методом множественного доступа с прослушиванием несущей и разрешением коллизий (конфликтов) (CSMA/CD - Carier Sense Multiple Access with Collision Detection).

Перед началом передачи рабочая станция определяет, свободен канал или занят. Если канал свободен, станция начинает передачу.

Ethernet не исключает возможности одновременной передачи сообщений двумя или несколькими станциями. Аппаратура автоматически распознает такие конфликты, называемые коллизиями. После обнаружения конфликта станции задерживают передачу на некоторое время. Это время небольшое и для каждой станции свое. После задержки передача возобновляется.

Реально конфликты приводят к уменьшению быстродействия сети только в том случае, если работает порядка 80-100 станций.

11.4.2. Метод доступа Arcnet

Этот метод доступа разработан фирмой Datapoint Corp. Он тоже получил широкое распространение, в основном благодаря тому, что оборудование Arcnet дешевле, чем оборудование Ethernet или Token -Ring. Arcnet используется в локальных сетях с топологией "звезда". Один из компьютеров создает пециальный маркер (сообщение специального вида), который последовательно передается от одного компьютера к другому.

Если станция желает передать сообщение другой станции, она должна дождаться маркера и добавить к нему сообщение, дополненное адресами отправителя и назначения. Когда пакет дойдет до станции назначения, сообщение будет "отцеплено" от маркера и передано станции.

11.4.3. Метод доступа Token-Ring

Метод доступа Token-Ring был разработан фирмой IBM и рассчитан на кольцевую топологию сети.

Этот метод напоминает Arcnet, так как тоже использует маркер, передаваемый от одной станции к другой. В отличие от Arcnet, при методе доступа Token-Ring имеется возможность назначать разные приоритеты разным рабочим станциям.

11.5. Аппаратное обеспечение локальных сетей

11.5.1. Аппаратура Ethernet

Аппаратура Ethernet обычно состоит из кабеля, разъемов, Т-коннекторов, терминаторов и сетевых адаптеров. Кабель, очевидно, используется для передачи данных между рабочими станциями. Для подключения кабеля используются разъемы. Эти разъемы через Т-коннекторы подключаются к сетевым адаптерам - специальным платам, вставленным в слоты расширения материнской платы рабочей станции. Терминаторы подключаются к открытым концам сети.

Для Ethernet могут быть использованы кабели разных типов: тонкий коаксиальный кабель, толстый коаксиальный кабель и неэкранированная витая пара. Для каждого типа кабеля используются свои разъемы и свой способ подключения к сетевому адаптеру.

В зависимости от кабеля меняются такие характеристики сети, как максимальная длина кабеля и максимальное количество рабочих станций, подключаемых к кабелю.

Как правило, скорость передачи данных в сети Ethernet достигает 10 Мбит в секунду, что достаточно для многих приложений.

Рассмотрим подробно состав аппаратных средств Ethernet для различных типов кабеля.

Толстый коаксиальный кабель, используемый Ethernet, имеет диаметр 0.4 дюйма и волновое сопротивление 50 Ом. Иногда этот кабель называют "желтым кабелем". Это самый дорогостоящий из рассматриваемых нами кабелей. Институт IEEE определил спецификацию на этот кабель - 10BASES.

На рис. 11.4 схематически изображена локальная сеть на основе толстого коаксиального кабеля.

Здесь приведена конфигурация сети, состоящей из двух сегментов, разделенным репитером. В каждом сегменте находятся 3 рабочие станции.

Каждая рабочая станция через сетевой адаптер (установлен на материнской плате компьютера и на рисунке не показан) специальным многожильным трансиверным кабелем подключается к устройству, называемому трансивером. Трансивер служит для подключения рабочей станции к толстому коаксиальному кабелю.

На корпусе трансивера имеется 3 разъема: два - для подключения толстого коаксиального кабеля, и один - для подключения трансиверного кабеля.

В таблице 11.1 перечислены устройства, необходимые для подключения рабочей станции к толстому коаксиальному кабелю.

К сожалению, длина одного сегмента ограничена, и для толстого кабеля не может превышать 500 метров. Если общая длина сети больше 500 метров, ее необходимо разбить на сегменты, соединенные друг с другом через специальное устройство - репитер.

На рисунке изображены два сегмента, соединенные репитером. При этом общая длина сети может достигать одного километра.

Между собой трансиверы соединяются отрезками толстого коаксиального кабеля с припаянными к их концам коаксиальными разъемами.

Ethernet на тонком коаксиальном кабеле

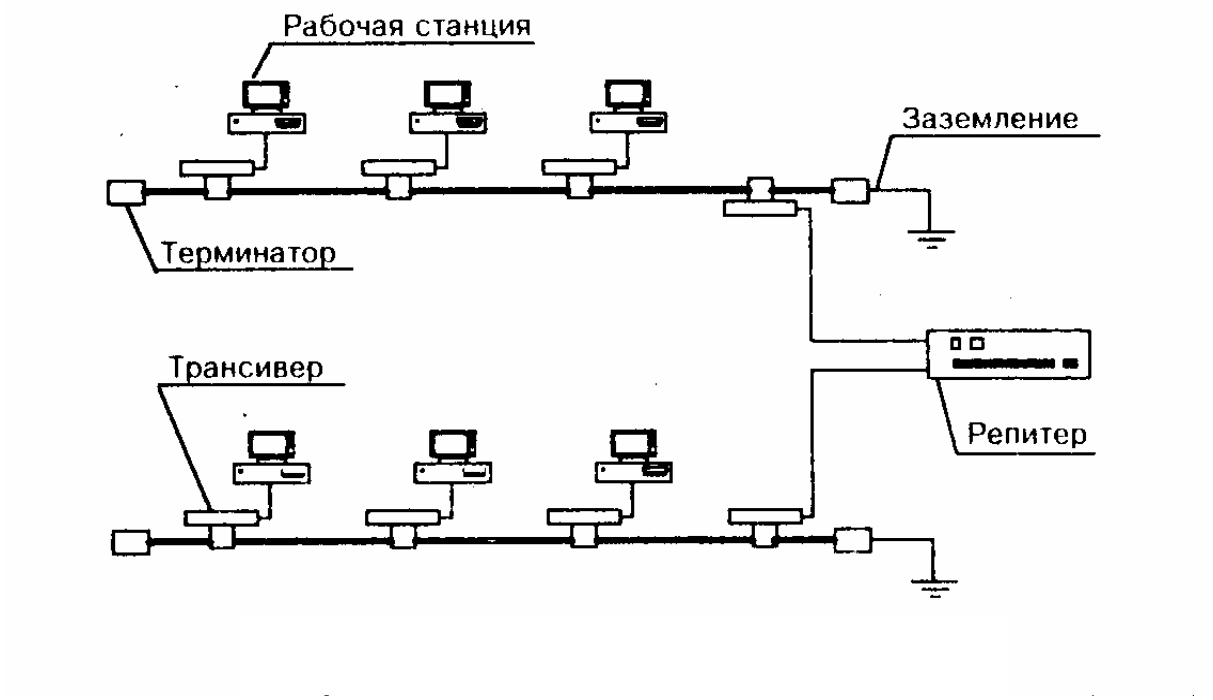


Рис. 11.4.

Таблица 11.1

Оборудование для подключения рабочей станции к толстому коаксиальному кабелю Ethernet

Сетевой адаптер компьютера	Вставляется в материнскую плату
Трансиверный кабель	Многожильный экранированный кабель, соединяет сетевой адаптер с трансивером
Трансивер	Соединяется трансиверным кабелем с сетевым адаптером, имеет два коаксиальных разъема для подключения к толстому кабелю разъемами.

На концах сегмента подключены специальные заглушки - терминаторы. Это просто коаксиальные разъемы, в корпусе которых установлен резистор с сопротивлением 50 Ом.

Корпус одного из терминаторов должен быть заземлен. В каждом сегменте сети можно соединять только один терминатор.

Существуют и другие ограничения кроме максимальной длины коаксиального кабеля.

Таблица 11.2

Ограничения для Ethernet на толстом кабеле

Максимальная длина сегмента	500 м
Максимальное количество сегментов в сети	5
Максимальная длина сети	2,5 км
Максимальное количество станций, подключенных к одному сегменту (если в сети есть репитеры, то они тоже считаются как рабочие станции)	100
Минимальное расстояние между точками подключения рабочих станций	2,5 м
Максимальная длина трансиверного кабеля	50 м

Кроме ограничения на длину сегмента существуют ограничения на максимальное количество сегментов в сети (и, как следствие, на максимальную длину сети), на максимальное количество рабочих станций, подключенных к сети и на максимальную длину трансиверного кабеля.

Однако в большинстве случаев эти ограничения не существенны. Более того, возможности толстого кабеля избыточны.

Итак, перечислим оборудование, необходимое для сети Ethernet на толстом кабеле:

N-коннектор

N-терминатор
N-Barrel-коннектор
N-терминатор с заземлением
DIX-коннектор
Трансивер

Тонкий коаксиальный кабель, используемый для Ethernet, имеет диаметр 0.2 дюйма и волновое сопротивление 50 Ом. Импортный кабель называется RG-58A/U и соответствует спецификации 10BASE2. Можно также использовать кабель РК-50, выпускаемый нашей промышленностью.

Сеть Ethernet на тонком кабеле существенно проще, чем на толстом.

Как правило, все сетевые адаптеры имеют два разъема. Один из них предназначен для подключения многожильного трансиверного кабеля, второй - для подключения небольшого тройника, называемого Т-коннектором. Т-коннектор с одной стороны подключается к сетевому адаптеру, а с двух других сторон к нему подключаются отрезки тонкого коаксиального кабеля с соответствующими разъемами на концах. При этом получается, что коаксиальный кабель подключается как бы непосредственно к сетевому адаптеру, поэтому не нужны трансивер и трансиверный кабель.

На концах сегмента должны находиться терминаторы, которые подключаются к свободным концам Т-коннекторов. Один (и только один!) терминатор в сегменте должен быть заземлен.

Сети на тонком кабеле имеют худшие параметры по сравнению с сетями на базе толстого кабеля (таблица 11.3). Но стоимость сетевого оборудования, необходимого для создания сети на тонком кабеле, существенно меньше.

Следует отметить, что некоторые фирмы выпускают адаптеры Ethernet, способные работать при длине сегмента до 300 метров (например, адаптеры фирмы 3COM). Однако такие адаптеры стоят дороже и вся сеть в этом случае должна быть сделана с использованием адаптеров только одного типа.

Таблица 11.3

Ограничения для Ethernet на тонком кабеле

Максимальная длина сегмента	185 м
Максимальное количество сегментов в сети	5
Максимальная длина сети	925
Максимальное количество станций, подключенных к одному сегменту (если в сети есть репитеры, то они тоже считаются как рабочие станции)	30
Минимальное расстояние между точками подключения рабочих станций	0,5 м

Как правило, большинство сетей Ethernet создано именно на базе тонкого кабеля.

Итак, перечислим оборудование, необходимое для сети Ethernet на тонком кабеле:

BNC-коннектор

BNC-терминатор

BNC-Barrel-коннектор

BNC-терминатор с заземлением

Т-коннектор

Некоторые (но не все) сетевые адаптеры Ethernet способны работать с кабелем, представляющем собой простую неэкранированную витую пару проводов (спецификация 10BASE-T). В качестве такого кабеля можно использовать обычный телефонный провод и уже имеющуюся в вашей организации телефонную сеть.

Сетевые адаптеры, способные работать с витой парой, имеют разъем, аналогичный применяемому в импортных телефонных аппаратах.

Для сети Ethernet на базе витой пары необходимо специальное устройство - концентратор. К одному концентратору через все те же телефонные розетки можно подключить до 12 рабочих станций. Максимальное расстояние от концентратора до рабочей станции составляет 100 метров, при этом скорость передачи данных такая же, как и для коаксиального кабеля, - 10 Мбит в секунду.

Достоинства сети на базе витой пары очевидны - низкая стоимость оборудования и возможность использования имеющейся телефонной сети. Однако есть серьезные ограничения на количество станций в сети и на ее длину.

11.5.2. Сетевой адаптер Ethernet

Вне зависимости от используемого кабеля для каждой рабочей станции необходимо иметь сетевой адаптер. Сетевой адаптер - это плата, которая вставляется в материнскую плату компьютера. Она имеет два разъема для подключения к сетевому кабелю.

Для Ethernet в стандарте ISA используется три вида сетевых адаптеров: 8-битовые, 16-битовые и 32-битовые. 8-битовый адаптер может вставляться в 8-битовый или 16-битовый слоты материнской платы и используется, главным образом, в компьютерах IBM XT IBM PC, где нет 16-битовых слотов. Иногда 8-битовые адаптеры используются для компьютеров IBM AT, если требования к скорости передачи данных не высоки. Для 16-битового адаптера необходимо использовать 16-битовый слот.

На компьютерах 80386 или 80486 имеет смысл использовать скоростные 32-битовые адаптеры, по крайней мере для тех станций, на которые приходится максимальная нагрузка.

Сетевые адаптеры могут быть рассчитаны на архитектуру ISA/EISA или Micro Channel. Первая архитектура используется в серии компьютеров IBM AT и совместимых с ними, вторая - в мощных станциях на базе процессоров 80486,

третья - в компьютерах PS/2 серии IBM. Конструктивно эти типы адаптеров отличаются друг от друга. Для ускорения работы на плате сетевого адаптера может находиться буфер. Размер этого буфера различен для адаптеров разных типов и может составлять от 8 Кб для 8-битовых адаптеров до 16 Кб и более для 16- и 32-битовых адаптеров.

Сетевые адаптеры Ethernet используют порты ввода/вывода и один канал прерывания. Некоторые адаптеры могут работать с каналами прямого доступа к памяти (DMA).

На плате адаптера может располагаться микросхема постоянного запоминающего устройства (ПЗУ) для создания так называемых бездисковых рабочих станций. Это компьютеры, в которых нет ни винчестера, ни флоппи-дисков. Загрузка операционной системы выполняется из сети, и выполнят ее программа, записанная в микросхеме дистанционной загрузки.

Перед тем как вставить сетевой адаптер в материнскую плату компьютера, необходимо с помощью переключателей (расположенных на плате адаптера) задать правильные значения для портов ввода/вывода, канала прерывания, базовый адрес ПЗУ дистанционной загрузки бездисковой станции.

Если длина сети превышает максимальную длину сегмента сети, необходимо разбить сеть на несколько (до пяти) сегментов, соединив их через репитер.

Конструктивно репитер может быть выполнен либо в виде отдельной конструкции со своим блоком питания, либо в виде платы, вставляемой в слот расширения материнской платы компьютера.

Репитер в виде отдельной конструкции стоит дороже, но он может быть использован для соединения сегментов Ethernet, выполненных как на тонком, так и на толстом кабеле, так как он имеет и коаксиальные разъемы, и разъемы для подключения трансиверного кабеля. С помощью этого репитера можно даже соединить в единую сеть сегменты, выполненные и на тонком, и на толстом кабеле.

Репитер в виде платы имеет только коаксиальные разъемы и поэтому может соединять только сегменты на тонком коаксиальном кабеле. Однако он стоит дешевле, и не требует отдельной розетки для подключения электропитания.

Один из недостатков встраиваемого в рабочую станцию репитера заключается в том, чтобы для обеспечения круглосуточной роботы сети станция с репитером также должна работать круглосуточно. При выключении питания связь между сегментами сети будет нарушена.

Функции репитера заключаются в физическом разделении сегментов сети и обеспечении восстановления пакетов, передаваемых из одного сегмента сети в другой.

Репитер повышает надежность сети, так как отказ одного сегмента (например, обрыв кабеля) не сказывается на работе других сегментов. Однако, разумеется, через поврежденный сегмент данные проходить не могут.

11.5.3. Аппаратура Arcnet

Для организации сети Arcnet необходим специальный сетевой адаптер. Этот адаптер имеет один внешний разъем для подключения коаксиального кабеля.

Каждый адаптер Arcnet должен иметь для данной сети свой номер. Этот номер устанавливается переключателями, расположенными на адаптере, и находятся в пределах от 0 до 255.

Таблица 11.4

Ограничения для сети Arcnet

Максимальная длина кабеля, который идет к активному концентратору	300 м
Минимальное расстояние между рабочими станциями, подключенными к одному кабелю	00,9 м
Максимальная длина сети по самому длинному маршруту	6 м
Максимальное расстояние между рабочей станцией и пассивным концентратором	30 м
Максимальное расстояние между активным и пассивным концентраторами	30 м
Максимальное расстояние между двумя активными концентраторами	600 м

Сетевые адаптеры рабочих станций через коаксиальный кабель с волновым сопротивлением 93 Ом подключаются к специальному устройству - концентратору. Возможно также использование неэкранированной витой пары.

Концентраторы бывают пассивными (Passive Hub) и активными (Active Hub). К одному концентратору (в зависимости от его типа) может подключаться 4,8,16 или 32 рабочих станций.

Ограничения для сети Arcnet приведены в таблице 11.4.

Достоинствами сети Arcnet являются низкая стоимость сетевого оборудования (по сравнению с Ethernet) и большая длина сети (до 6 км). Однако низкая скорость передачи данных, составляющая 2.44 Мбит в секунду, ограничивает применение сети Arcnet.

11.5.4. Аппаратура Token-Ring

Что касается сети Token-Ring, то ее название может ввести вас в заблуждение. Топология этой сети больше похожа на топологию звезды, чем на топологию кольца. Вместо того чтобы, соединяясь друг с другом, образовывать кольцо, рабочие станции Token-Ring подключаются радиально к концентратору типа 8228 производства IBM. Правда, концентраторов может быть несколько, и в этом случае концентраторы действительно объединяются в кольцо через специальные разъемы.

Однако если используется один концентратор, то объединяющие разъемы можно не закольцовывать. Скорость передачи данных в сети Token-Ring

может достигать 4 или 16 Мбит в секунду, однако стоимость сетевого оборудования выше, чем для сети Ethernet. Кроме того, существуют и другие ограничения (см. таблицу 11.5.).

Таблица 11.5

Ограничения для сети Token-Ring

Максимальное количество концентраторов типа 8228 в сети	12
Минимальное количество рабочих станций в сети	96
Максимальная длина кабеля между двумя концентраторами	45 м
Максимальная длина кабеля, соединяющая все концентраторы в сети	120 м

Как видно из этой таблицы, сети Token-Ring не рассчитаны на большие расстояния. Все компьютеры должны быть расположены на одном или двух этажах здания. Более высокая стоимость оборудования по сравнению с Ethernet, дополнительно уменьшает его частое применение.

Литература

1. Нешумова К.А. Электронные вычислительные машины и системы. М.: Высшая школа, 1989.
2. Морозевич А.Н. МикроЭВМ, микропроцессоры и основы микропрограммирования. М.: Высшая школа, 1990.
3. Толковый словарь по вычислительным системам./ Под.ред. Иллингурова В.:Пер. с англ. М.: Машиностроение, 1990.
4. Вычислительные системы, сети и телекоммуникации: учебник/ А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко; Под ред. А.П. Пятибратова. -М.: Финансы и статистика, 1998.
5. Калабеков Б. Цифровые устройства и микропроцессорные системы. Учебник. Гор. лин.- Т:, 1999.

Подписано в печать _____ Заказ _____ Объем_____
Тираж _____Формат 60x84. 1/16. Печать оперативная.
Типография_____Роапринт_____
Московская государственная академия приборостроения и информатики,
Москва 107076, ул. Стромынка 20.